RSTS/E V06B-02

JANUARY 1977

Release Notes

DEC-11-ORINC-A-D

DIGITGRAPH
P. O. Box 4249
Tucson, Arizona 85717

TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION
MAYNARD, MASSACHUSETTS

DIGITAL	DECsystem-10	MASSBUS
DEC	DECtape	OMNIBUS
PDP	DIBOL	OS/8
DECUS	EDUSYSTEM	PHA
UNIBUS	FLIP CHIP	RSTS
COMPUTER LABS	FOCAL	RSX
COMTEX	INDAC	TYPESET-8
DDT	LAB-8	TYPESET-11
DECCOMM		

SOFTWARE SERVICES

Copyright 1976 DIGITAL EQUIPMENT CORPORATION

First Printing, January 1977

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may only be used or copied in accordance with the terms of such license.

Digital Equipment Corporation assumes no responsibility for the use or reliability of its software on equipment that is not supplied by Digital.

Copyright © 1977, Digital Equipment Corporation

DECUS MEMBERSHIP

DECUS is a voluntary, nonprofit users' group, supported in part by Digital Equipment Corporation. DECUS and the RSTS SIG provide valuable newsletters, symposia, and library offerings to their members.

You can apply for membership (with no voting privileges) or can have your installation apply for membership and become its official delegate.

DECUS CHAPTER OFFICES

Australia/NZ:
DECUS Australia
P.O. Box 491
Crows Nest, N.S.W. 2065
Australia

Canada:
DECUS Canada
P.O. Box 11500
Ottawa, Ontario K2H 8K8
Canada

Europe/Middle East:
DECUS Europe
Case Postale 340
1211 Geneva 26
Switzerland

U. S. and All Others:*
DECUS United States
146 Main Street
Maynard MA 01754
U. S. A.

*Bob Branton
R.La Bar & Associates
2750 Northaven Road, Suite 102
Dallas TX 75229
Tel: (214) 243-3451

CONTENTS

1.0 Introduction

2.0 Differences Between RSTS/E V06B-02 and V06A-02
2.1 New Processor Support
2.2 New Device Support
2.2.1 DZ11 Asynchronous Multiplexor
2.2.2 DJ11 Asynchronous Multiplexor
2.2.3 RK05F Disk Drive
2.2.4 RK06 High Performance Disk Cartridge Drive
2.2.5 RP05 and RP06 High Capacity, High Performance Disk Drives
2.2.6 LA180 DECprinter I
2.2.7 Magnetic Tape Support
2.3 System Generation
2.4 Initialization Code
2.5 Monitor and File Processing
2.6 Terminal Service
2.7 BASIC-PLUS Language and Run-Time System
2.8 RSTS/E Utility Library Programs
2.8.1 RSTS/E System Utilities Package
2.8.2 RSTS/E Data Manipulation Package
2.8.3 RSTS/E Error Control Package
2.8.4 RSTS/E Backup Package
2.8.5 RSTS/E Spooling and Operator Services Package
2.8.6 RSTS/E Run-Time System Support Package
2.8.7 RSTS/E User Environment Test Package
2.9 Documentation
2.10 Old Problems Corrected
2.10.1 RSTS/E Monitor
2.10.2 BASIC-PLUS Language and Run-Time System
2.10.3 Commonly Used System Programs
2.10.4 Run-Time Systems and Support
2.11 Upgrading from V06A-02 to V06B-02
2.11.1 Creating a V06B-02 System on a V06A Nonsystem Disk

3.0 Contents of RSTS/E V06B-02 Distribution Kits

4.0 Notebook Maintenance
4.1 RSTS/E COMPONENT.SUB-COMPONENT Assignments

5.0 Software Performance Reports
5.1 Software Performance Report Guidelines

6.0 Patching Procedures
6.1 Monitor Patching
6.2 Commonly Used System Program Patching

7.0 Release Notes and Software Dispatch Articles
7.1 Cumulative Index of All Articles
7.2 Commonly Used System Programs Status Report

APPENDIX A
SYSGEN Editor Manual

1.0 Introduction

The RSTS/E V06B-02 Release Notes establish a maintenance notebook for documentation corrections, published software problems and solutions, and programming notes. Subordinate software information is included as well. The Software Dispatch, which is sent for one year after each Category A or B software installation, is designed as an insert. It is addressed to the software contact established by the DIGITAL field office. After one year, the customer may purchase the Binary Program Update Service.

Anyone who is about to generate RSTS/E V06B-02 must incorporate "Mandatory" and appropriate "Optional" patches as part of the system generation procedure. The patches are presented in the Cumulative Index with a compilation of the version or edit levels for all of the Commonly Used System Programs (CUSPs). These indices are updated regularly in the Software Dispatch.

For further information on RSTS/E documentation, refer to the RSTS/E Documentation Directory.

2.0 Differences Between RSTS/E V06B-02 and V06A-02

RSTS/E V06B-02 introduces new hardware support and several new software features. Improvements were made in the internal structure of the monitor, overlay, initialization code and BASIC-PLUS programs. The system generation procedures were re-implemented to provide a RSTS/E based generation process and to eliminate the DOS dependency. Support for BASIC-PLUS II and RMS-11 have been included as well as facilities for supporting DECNET/E when it becomes available.

All known V06A-02 problems have been corrected. The manual set has been revised to reflect the new hardware changes and software corrections. Several new additions were made to the manual set as well as a complete revision of several of the documents.

Section 2.11 presents guidelines for installations upgrading from V06A-02 to V06B-02.

2.1 New Processor Support

No new processors are supported under V06B-02. Support for up to 4 million bytes of memory for the 11/70 has been included. The 11/34 FPP compatible floating-point option is also supported in V06B-02.

2.2 New Device Support

Several devices introduced during the period after V06A-02 was released have been included in the supported device category for V06B-02.

2.2.1 DZ11 Asynchronous Multiplexor

The DZ11 Asynchronous Multiplexor is a PDP-11 peripheral device that provides as many as eight communications channels between a variety of devices and the PDP-11 UNIBUS. The DZ11 operates full-duplex on asynchronous serial data input from a modem or terminal, converting the data stream to the parallel format of the UNIBUS. Conversely, parallel data from the UNIBUS is serialized by the DZ11 for transmission to the modem or terminal. Both EIA and 20 ma. current-loop interfacing is available as well as control capability for BELL 103A or equivalent modems.

The DZ11 features flexible control of communications parameters including baud rate, character length, number of stop bits for each line, odd or even parity for each line, and transmitter/receiver interrupts. Additional features include limited data set control, zero receiver baud rate, break generation and detection, and silo buffering of received data.

Each DZ11 provides operation for up to eight serial lines. Up to sixteen DZ11 multiplexors may be included in a RSTS/E installation.

The DZ11 multiplexor is supported under RSTS/E V06B-02 through the normal terminal service. All standard terminal characteristics except SPLIT SPEED can be used for terminals attached to DZ11 lines. SPLIT SPEED operation is not available because the DZ11 hardware permits only a single speed selection which applies to both transmit and receive on a given line.

2741 terminal support is also available on DZ11 lines.

2.2.2 DJ11 Asynchronous Multiplexor

RSTS/E V06B-02 also supports the DJ11 serial asynchronous multiplexor. Since DJ11 line characteristics are not programmable in the multiplexor, the terminal characteristics controllable through TTYSET commands are the same as for a single line interface such as the DL11. No modem control is provided in the DJ11 hardware.

2.2.3 RK05F Disk Drive

RSTS/E V06B-02 now supports the RK05F disk drive as a member of the overall RK05 disk structure.

If RK05 distribution is to be used, at least one RK05J removable cartridge drive must be included in the system configuration. The minimum is two RK05J drives if there are no RK05F or other disk devices included in the configuration.

2.2.4 RK06 High Performance Disk Cartridge Drive

The RK06 disk drive is DIGITAL's new medium capacity disk drive designed for small to mid-range PDP-11 configurations. It features a storage capacity of 27104 blocks (512 bytes/block) in a compact cartridge design and uses technology similar to the RP04/RP05/RP06 disk drive family to provide reliability and high performance.

The RK06 interfaces to the PDP-11 through the RK611 controller. Up to eight drives may appear on a single RSTS/E system.

The RK06 disk drive is identified in RSTS/E by the DMn: disk device name. The RK06 may be used as the system disk, as part of the public structure, or as a private disk. RSTS/E V06B-02 is available for distribution on RK06 media. For systems with no magnetic tape, and no RK05 disk drives, at least two RK06 drives must be configured.

2.2.5 RP05 and RP06 High Capacity, High Performance Disk Drives

The RP05 and RP06 disk drives offer the same high level of performance and broad range of features currently provided in the RP04 disk drive. The two new drives use the same PDP-11 device controllers as the RP04. All three drive types can be intermixed on a single PDP-11 configuration.

The RP05 capacity is the same as that of the RP04 disk drive. The RP05, however, is upgradable in the field to double its capacity to make it an RP06. Thus, RSTS/E users can now have over 320K blocks of storage/drive in their on-line disk sub-systems using the RP06 capability.

RSTS/E V06B-02 supports the RP04/RP05/RP06 drives as part of the same basic structure. All three devices are referred to by the DBn: disk device name. RSTS/E is not distributed on any of these media.

2.2.6 LA180 DECprinter I

The LA180 DECprinter I low-cost line printer is supported by RSTS/E V06B-02. When connected to the PDP-11 UNIBUS through a parallel interface, the LA180 is made known to the RSTS/E monitor as an LP11 printer, with the desired assortment of options. In this regard, the printer is handled as a standard LP11 printer.

When connected to RSTS/E as an LP11 equivalent printer, the hardware forms length capability of the LA180 is supported only when used under the control of the RSTS/E Spooling package. For details of this feature and its control, consult the Release Note Articles on the Spooling package, the appropriate section of the RSTS/E System Manager's Guide, and the LA180 DECprinter I User's Manual.

The serial interfaced version of the LA180 (henceforth called the LA180S) is supported (with some restrictions) under V06B-02 as a receive-only printing terminal through the Spooling package. For details of this support, consult the Release Note Articles on the Spooling package, the appropriate section of the RSTS/E System Manager's Guide, the LA180 DECprinter I User's Manual, and the LA180S Buffered Serial Interface User's Manual.

2.2.7 Magnetic Tape Support

RSTS/E V06B-02 now supports the following 7 and 9-track tape drives:

TS03	9-track only
TU10	7 and 9-track
TU16	9-track only
TU45	9-track only

On a given RSTS/E system, all TS03 and TU10 tape drives are treated as part of the same device class, and all TU16 and TU45 tape drives are grouped together. Up to eight units of each class may be installed on the same RSTS/E installation.

2.3 System Generation

System generation is now done under a pre-built RSTS/E V06B-02 monitor instead of DOS/BATCH. All system generation media are RSTS/E file structured. While similar in flow to the V06A-02 system generation procedure, the V06B-02 System Generation Manual (DEC-11-ORGNA-B-D) should be read in full prior to generating a system.

2.4 Initialization Code

The system initialization code has been completely re-written. The System Generation Manual (DEC-11-ORGNA-B-D) and System Manager's Guide (DEC-11-ORSMD-B-D) should be carefully read to become familiar with the new options and procedures.

The new INIT does a complete configuration check every time it is loaded (booted). Any non-existent device is automatically disabled. In addition, all devices are forced to interrupt and their vector locations and priority levels are determined. Devices which won't interrupt, interrupt at too high a priority level, or interrupt to the same vector as another device are also automatically disabled.

2.5 Monitor and File Processing

The internal system tables and data structures were completely revised and re-organized to improve the internal memory utilization in the monitor and to implement various standards throughout the monitor. This permitted the removal of a large number of inconsistencies between various structures which developed as RSTS/E evolved.

A standard form for the Device Data Block (DDB) has been developed and all device drivers now use this format. V06A-02 programs which PEEKed at locations in DDB's may have to be modified to account for the new structure.

All internal system table structures have been revised. For control of jobs, the Job Data Block format has changed and a second block of information has been added.

The memory control structure was changed to allow for up to 4 million bytes of memory for 11/70 systems. The structure was also modified as part of upgrading Run-Time System support in V06B-02.

Miscellaneous areas of further improvement were Run-Time System handling and support, general device control, standardized device naming structures, and device handler support.

A central device time-out facility was implemented in the monitor. All system device drivers use that facility in controlling their devices.

In the disk control section of the monitor, an improvement was made which allows both device type and unit number to be selected at bootstrap time (V06A-02 allowed only the unit number to be changed).

Improvements in the control of the monitor memory mapping registers were implemented. These changes allow several monitor segments (FIP, EMT services, I/O driver routines, 2780 handler, and future network code) to be mapped outside of the lower 20K words of physical memory. These changes also allow the amount of monitor required to be permanently mapped into the first 20K words of memory to be minimized and allows greater flexibility in configuring the proper number of small buffers for optimal system performance.

Generalized system logical names have been implemented. V06A-02 provided only one system logical name: SY0: for the system disk. V06B-02 allows up to thirty logical names, each of which can contain up to six characters. Each name can be associated with both a device and an account; for example, the name LIB: could be used to represent DK1:[1,100] in any file specification. These associations are established at system startup using the UTILTY program.

The send/receive message facility of V06A-02 was completely re-written to provide facility for transmitting messages within a system or, in the future, through DECNET. A new set of system function SYS calls has been provided to interface to the new facility, with the old form SYS calls retained in the interest of compatibility. SEND/RECEIVE messages may now contain up to 532 bytes of information. Provision has been made for selective receive of messages based on sending job number as well as providing for selective packing of message buffers during reception.

The Concise Command Language (CCL) facility of the BASIC-PLUS Run-Time System was moved into the RSTS/E monitor. Commands have been added to allow the on-line addition and deletion of CCL commands. Monitor small buffers are now used for storage of CCL information. Each CCL has a unique abbreviation specified by the command installer. Additionally, the specified CCL commands are available to all Run-Time Systems, such as FORTRAN IV.

For file processing, four additional file OPEN modes have been added. The first, known as "read-only", allows a file to be OPENed for read privileges only, even if write privileges would ordinarily be granted.

The second OPEN mode, known as "read-regardless", now allows OPENing of files, even if they are already OPEN in UPDATE mode. Any record in the file may then be read, even if it is locked.

The third OPEN mode allows for the creation of contiguous files. Contiguous files do not require "window turns" for random access processing and result in improved system performance in data base applications. These files may not be extended after creation.

The fourth OPEN mode keeps a sequential output file's length information always updated in the file's directory. This ensures that all of the file's data can be recovered after a system crash/power fail.

A new device, called the NULL device, is available to all users. This device returns an immediate end-of-file error on reads (GETs) and serves as a "bit bucket" on writes (PUTs).

Update mode has been enhanced to allow the locking of up to fifteen blocks. In BASIC-PLUS, only contiguous blocks may be locked. Multiple scattered block locks can be achieved using the RMS-11 Package.

A central file name string scan facility has been added to the monitor. This central facility will be used by all Run-Time Systems to provide for a uniform method of scanning and parsing file specification strings for RSTS/E files. As networking is introduced, the importance of this central facility increases.

The support of Run-Time Systems now includes an automatic loading capability which is invoked when the user "runs" the Run-Time System. This facility allows automatic selection of the location of the Run-Time System in memory and prevents conflicts between Run-Time Systems competing for memory space.

An improvement to the handling of privilege by privileged programs has been provided whereby a program may request the system to "drop" its privileges temporarily, such that the program can re-gain them at a later time. This allows a privileged program to use the file protection features of RSTS/E at file OPEN time, after which the program may re-gain privileges if it needs them.

The swapping file structure was modified to allow for the addition/removal of system swap files during timesharing operations. The placement of swap files may now be made on any RSTS/E disk system with a file structure or on other disks in non-file structured mode.

Lastly, the error logging sections of the monitor and all device drivers were revised. The amount of information logged for each error was increased, and a new set of error control utility programs provided.

2.6 Terminal Service

The Terminal Service was upgraded to new monitor conventions and support was added for DZ11 and DJ11 multiplexors.

Additionally, a new operation mode, called "echo control" mode, was implemented to facilitate development of "screen form" types of input procedures. This vastly simplifies the implementation of CRT-oriented, "fill in the blanks" terminal application programs.

Each keyboard in the system may now have one extra, user-specified delimiter character assigned to it, performing the same input text termination functions as <cr>, <lf>, <esc> and <ff>. For example, the character <tab> could be used in conjunction with the "echo control" feature to provide an improved data entry utility.

2.7 BASIC-PLUS Language and Run-Time System

Several changes were made to the BASIC-PLUS language and Run-Time System to improve old functionality and add new features.

The LENGTH command now reports both the current private memory size along with the maximum size. For example:

```
LENGTH
5(8K) of memory used
```

will appear for a program using 5K of memory, with a maximum allowed size of 8K.

To avoid having to copy data from one I/O buffer into another when copying files, an "alternate I/O buffer" capability has been added to GET/PUT operations in BASIC-PLUS. The general form of the GET statement which uses an alternate I/O buffer is as follows:

```
GET #SWAP%(B%)+I% [,RECORD R%]
```

The input file is OPEN on channel I%. If B% = 0%, then the GET reverts to the standard operation using the buffer associated with channel I%. If B% <> 0%, then the resulting data fetched with the GET will be placed in the buffer associated with channel B%, where 1% <= B% <=12%.

The form of the PUT statement using an alternate I/O buffer is as follows:

```
PUT #SWAP%(B%)+I% [,RECORD R%] [,COUNT C%]
```

As with the GET statement, B% = 0% reverts to the standard GET operation. If B% <> 0%, then the data will be written from the buffer associated with channel B%.

For both the GET and PUT statements, channel B% must be OPEN. The new RSTS/E null device can be very useful in this case by providing buffer space for the OPEN channel without requiring an actual file to be OPEN on disk.

A sample program segment for copying two files whose record sizes are the same follows.

```
2010      OPEN "FOO.OLD" FOR INPUT  AS FILE 1%
        \ OPEN "FOO.NEW" FOR OUTPUT AS FILE 2%
                ! OPEN BOTH OLD AND NEW FILES

2020      GET #1%
        \ PUT #SWAP%(1%)+2%
        \ GO TO 20
                ! GET RECORD FROM INPUT.
                ! OUTPUT RECORD, USING INPUT CHANNEL BUFFER
                ! CLOSED LOOP.  LOOP IS EXITED VIA END-OF-FILE
                ! ERROR (ERR=11)
```

The rules concerning the specification and processing of the RECORDSIZE modifier for the file OPEN statement have been modified for V06B-02. Assuming a RECORDSIZE value of S%, and a RSTS/E default value for RECORDSIZE of D% (consult the RSTS/E System User's Guide for default RECORDSIZE values for peripheral devices), the rules for processing the RECORDSIZE are as follows:

a) Force S% to be an even quantity (i.e. S% = S% AND -2%)

b) The RSTS/E monitor assumes a default value for D% according to the associated device type of the file being OPENed.

c) If S% = 0%, then use the RSTS/E default value of D%.

d) If S% > 0% (positive value), then:

   d1) Use the user supplied value of S% if S% > D%

   d2) Use the RSTS/E default value D% if S% <= D%.

e) If S% < 0% (negative value), first remove the sign bit (i.e. S% = S% AND 32767%). Then:

   e1) If the resulting S% = 0%, then use the RSTS/E default value of D%.

   e2) If the resulting S% <> 0%, then use the resulting value of S%.

As can be seen, all of the rules, except e2, are the same in V06B-02 as they were in V06A-02. Rule e2 is the new variation provided in V06B-02.

As an example of the new RECORDSIZE feature, statement 2010 of the previous GET/PUT example is re-written below:

```
2010      OPEN "FOO.OLD" FOR INPUT  AS FILE 1%
        \ OPEN "FOO.NEW" FOR OUTPUT AS FILE 2%,
                RECORDSIZE (32767%+1%+2%)
                        ! OPEN BOTH OLD AND NEW FILES
```

By using RECORDSIZE (32767%+1%+2%) on the output file OPEN, the associated buffer on channel 2% is established at a length of two bytes. This means that 510 bytes of data space would be saved while accomplishing the same copy operation. A modified version of the NUM$ function, called NUM1$, has been added to the BASIC-PLUS language. The NUM1$ function performs just like NUM$ except that:

No spaces are returned
The E format is never returned

The NUM$ function is still available.

An optional package for doing arithmetic operations on numeric data stored in character strings is available in V06B-02. String arithmetic may be included in BASIC-PLUS by answering the appropriate question during the system generation process.

The string arithmetic package provides the following functions:

```
SUM$     A$ = SUM$(B$,C$)        (Addition)

DIF$     A$ = DIF$(B$,C$)        (Subtraction)

PROD$    A$ = PROD$(B$,C$,P%)    (Multiplication)

QUO$     A$ = QUO$(B$,C$,P%)     (Division)

PLACE$   A$ = PLACE$(B$,P%)      (Truncation)

COMP%    T% = COMP%(A$,B$)       (Comparison)
```

where A$, B$, and C$ are strings of digits and P% is a precision length value. See the BASIC-PLUS Language Manual for further details on how these functions are used.

Error messages returned by the BASIC-PLUS Run-Time System have been improved as follows:

- If necessary, a <cr>-<lf> is output to restore the left-hand margin of the terminal

- Error messages begin with either "?" or "%" prefixed to the message. If the message is informational only, neither prefix character will appear.

These "?" (FATAL) and "%" (WARNING) designations are utilized by the BATCH facility for conditional control of batch job streams.

A new operating mode has been added to the BASIC-PLUS Language and Run-Time System which allows the following extensions of the language:

- Variable names of up to 30 characters are allowed. These names consist of a leading alpha character and up to 29 additional characters (alphanumeric and "."). In addition, a name may have an optional "FN" prefix (for functions) and/or either of the "%" or the "$" suffix.

- Because of the long variables name feature, spaces and tabs in the language and command syntax of BASIC-PLUS are significant. For example, FOR I = S TO P is now required, as is RUN $PIP.

- A line ending in

  &[<space/tab> sequence]<cr>

  will cause the next line to be treated as a continuation line (similar to the V06A-02 <lf> convention).

- If a line ends with

  !<comment not including "!">&[<space/tab>]<cr>

  the comment will be treated as an in-line comment with continuation of the program text on the next line.

This new operation mode is called the "extend" mode of BASIC-PLUS. Language and system commands allow for switching between the old compatibility mode (NO EXTEND) and the new mode (EXTEND). The EXTEND/NO EXTEND commands can be used in immediate mode or in BASIC-PLUS programs. If a mode change command appears in a program, its effect is local to the program. If the command is typed in immediate mode, the effect is permanent until the next immediate mode change. In either of the two operating modes of BASIC-PLUS, there is now no restriction on the total number of characters which can occur in a given BASIC-PLUS language statement. This means that the BASIC-PLUS programmer is free to format a program for readability; only restrictions in the BASIC-PLUS language remain.

The STATUS variable provided in BASIC-PLUS has been changed with regard to results returned after performing a "file name string scan" (see the System Programmer's Manual on system (SYS) functions). The old meaning for bit 12 (value 4096%) has been changed to mean "device accepts modifiers". The following table shows the meanings of the STATUS result depending on the returned values of bit 12, bit 13 and bit 15:

Bit 15	Bit 13	Bit 12	
100000	040000	010000	
-32768	16384	4096	Meaning
0	0	0	Block sequential, no modifiers
0	0	1	Block sequential, modifiers (Note 1)
0	1	0	Byte sequential, no modifiers
0	1	1	Byte sequential, modifiers (Note 2)
1	0	0	Block random, no modifiers
1	0	1	Block random, modifiers (Note 1)
1	1	0	** Impossible **
1	1	1	** Impossible **

Note 1: This case can not be handled by BASIC-PLUS.
Note 2: This case is handled by using the RECORD modifier value on GETs, PUTs, and/or PRINTs to the device in question.

Several changes have been made to the non-privileged system function calls (SYS).

The system function calls SYS(CHR$(x%)), where x = 0, 1, 2, 3, 4, and 11, have been changed to the following general format:

```
SYS(CHR$(x%)[+CHR$(C%)[+CHR$(K%)]])
```

where:

a) No C% or K% means use job's keyboard (KB:)
b) C% and no K% means use the terminal on channel C%
c) C% and K% both means use terminal keyboard # K% which is a multiple service terminal on channel C%

SYS call 5% has been modified to cause the following actions:

a) Do not "wipe out" program
b) Do not return to private Run-Time System
c) Set up to allow a CONTinue command to be typed
d) Do not issue a prompt message ("Ready")

SYS call 9% has been modified to cause the following actions:

a) Close all channels without cleaning up partial buffers
b) Always "wipe out" program
c) Always return to private Run-Time System
d) Tell the Run-Time System to issue a prompt

SYS call 9% has one additional format:

```
SYS(CHR$(9%)+CHR$(N0%)+CHR$(N1%)+CHR$(N2%)+CHR$(N3%))
```

The actions invoked by this form of SYS call 9% are:

a) Close all channels without cleaning up partial buffers
b) Always "wipe out" program
c) Return to the Run-Time System whose RTS name is encoded in values N0% through N3% in RAD50 format
d) Furthermore, establish the indicated Run-Time System as the user's private default Run-Time System
e) Tell the Run-Time System to issue a prompt

The user should read the V06B-02 manuals for more information on the use and control of Run-Time Systems.

As part of the new "echo control" mode features of the RSTS/E terminal service, a new SYS call 11% has been provided to cancel all currently pending input ("type ahead") This SYS call permits clearing of terminal input buffers prior to declaring a field on the terminal; thus, any characters which had been typed, but not echoed, will not overwrite any previously written characters on the terminal display.

The privileged SYS call -21% has been modified as follows. The form

```
SYS(CHR$(6%)+CHR$(-21%))
```

causes a permanent dropping of privileges for the calling program.

The form

```
SYS(CHR$(6%)+CHR$(-21%)+CHR$(255%))
```

causes the program to lose its privileges, but does so in a way that they can be re-gained at a later time by the SYS call

```
SYS(CHR$(6%)+CHR$(-21%)+CHR$(0%))
```

Existing BASIC-PLUS programs need not be recompiled to run under the V06B-02 BASIC-PLUS. However, BASIC-PLUS has been upgraded to provide better handling of integer and floating point constants, so some size reduction and/or speed enhancement may be achieved by recompiling.

2.8 RSTS/E Utility Library Programs

Rather than describe the entire set of Commonly Used System Programs (CUSPs) as a single library, the programs are grouped as a set of packages which are included in each RSTS/E Distribution kit. The packages are listed as follows:

RSTS/E System Utilities Package
RSTS/E Data Manipulation Package
RSTS/E Error Control Package
RSTS/E Backup Package
RSTS/E Spooling and Operator Services Package
RSTS/E Run-Time System Support Package
RSTS/E User Environment Test Package

The changes in programs in each of these packages is described below, along with brief descriptions of new programs being introduced with the V06B-02 release. Some changes occurred in nearly all of the programs in an effort to implement aspects of a BASIC-PLUS coding standard.

2.8.1 RSTS/E System Utilities Package

The RSTS/E System Utilities Package contains the following:

BUILD	DIRECT	DISPLY	DSKINT
GRIPE	INIT	INUSE	LOGIN
LOGOUT	MONEY	ODT	PLEASE
PRIOR	QUOLST	REACT	REORDR
SHUTUP	SYSCAT	SYSTAT	TALK
TTYSET	UMOUNT	UTILTY	UTILT1

BUILD

The BUILD program now allows libraries to be built onto some specified disk instead of the system disk and/or into a library account other than [1,2].

DIRECT

The DIRECT program has been extended to display more information about each file in a /S (slow) directory listing. Specifically, the name of the Run-Time System by which the file was created and any attribute information are shown.

DISPLY (VT05, VT50(VT52))

DISPLY is the base program used in conjunction with VT05.DPY and VT50.DPY to form programs for dynamic display of system status information on VT05, VT50 and VT52 terminals. The modules VT05.DPY and VT50.DPY are appended to DISPLY to form the programs VT5DPY and VT50PY which output to VT05 and VT50 or VT52 terminals, respectively. The system status display conforms to that of the SYSTAT utility program, within limitations on available screen space for display.

New commands added to the DISPLY base program allow sleeping jobs to be included/excluded from the job list and provide for display of each job's Run-Time System to be displayed instead of the job's name.

The VT05 control module functions are unchanged from Version V06A-02. The VT50 control module was modified to provide several program startup switches which permit the user to specify use of VT50 and VT52 features, such as 24 lines/screen, direct cursor addressing, hardware tab functions, and echo control.

DSKINT

The DSKINT program was re-written to permit it to initialize all disk types supported by RSTS/E. Additionally, any RSTS/E disk can now be re-initialized by DSKINT. In V06A-02, no re-initialization was possible, nor were any disk types other than RK05 supported.

The DSKINT program can not format a disk, nor can it perform pattern checks to locate bad blocks. If the disk being initialized is formatted, and contains a valid RSTS/E structure, DSKINT will retain bad block data if it is present. If the disk is formatted, but not necessarily in the RSTS/E structure, the bad block data will not exist.

During initialization, the user may specify the placement of the storage allocation table file (SATT.SYS), for example, at the mid point of the disk being initialized.

GRIPE

The GRIPE program functions in V06B-02 as it did in V06A-02.

INIT

The INIT program now has a built-in time-out interval of 30 seconds when the command file name is requested. If no reply has been given in that time, INIT automatically uses START.CTL. When re-attaching to a terminal, INIT now uses the REATTACH SYS call instead of going through a LOGIN sequence. Commands have been added to INIT to ADD SWAP files as well as OVERLAY and ERROR system files. (See System Manager's Guide.)

INUSE

The INUSE program functions in V06B-02 as it did in V06A-02.

LOGIN

The default LOGIN SWAP MAX value has been changed. In V06A-02, the default value was 8K. In V06B-02, this value has been increased to 28K.

Individual installations should re-evaluate their requirements with regard to default setup of jobs at LOGIN time.

LOGOUT

LOGOUT now deletes all files that match ????nn.TMP (where nn is the job number); it previously deleted only TEMPnn.TMP files.

MONEY

The MONEY program functions in V06B-02 as it did in V06A-02.

ODT

The ODT program functions in V06B-02 as it did in V06A-02.

PLEASE

PLEASE has been re-written in V06B-02 as part of the Spooling and Operator Services Package. If the Operator Service program OPSER is not running, messages input to PLEASE will be sent to the system console terminal (KB0:) as in V06A-02. If OPSER is present, PLEASE may be used to send either messages or commands to OPSER. Messages are printed by OPSER on some system designated terminal, known as the Operator Services Console (OSC), while commands are passed to OPSER for interpretation and execution.

PRIOR

The PRIOR program functions in V06B-02 as it did in V06A-02.

QUOLST

The QUOLST program functions in V06B-02 as it did in V06A-02.

REACT

The REACT program functions in V06B-02 as it did in V06A-02.

REORDR

The REORDR program was re-written to conform to the coding standards specified for RSTS/E CUSPs. Additionally, the program now permits the sorting of files in a User File Directory (UFD) according to either creation date/time or date of last access, with sorting in either ascending or descending order on the parameter(s) chosen.

SHUTUP

The SHUTUP program was re-written completely to provide a single program which would be used to shutdown the RSTS/E system under all operating conditions. The program functions appropriately whether the Spooling Package is in use or not.

Warning messages, detailing the remaining minutes until the shutdown procedure begins, now appear at intervals whose lengths decrease as the shutdown time approaches, resulting in fewer messages being broadcast. As SHUTUP runs, it performs its own system consistency checks and attempts to shutdown in the fastest possible way.

SHUTUP will operate in cooperation with the Operator Services Program to close down spooling operations if desired by the operator. SHUTUP and OPSER provide for either an immediate shutdown or a controlled shutdown, where the spoolers finish the job they are currently working on.

SHUTUP now takes care of removing all Run-Time Systems present on the system, removing SWAP files from usage, and terminating the error logging process. When shutdown is finished, the RSTS/E system initialization code INIT is re-booted so that the system manager may proceed as desired.

SYSCAT

The SYSCAT program functions in V06B-02 as it did in V06A-02.

SYSTAT

Several features have been added to SYSTAT in the V06B-02 release. New options available to the user are as follows:

Option	Purpose
/A	list attached jobs
/M	list message receiver status
/N	list jobs in non-privileged accounts
/P	list jobs in privileged accounts
/U	list detached jobs (replaces the /DET option)
/n,*	list all jobs for project number n

Any combination of the basic options (A, B, D, F, M, N, P, R, S and U) is permissible. A minus "-" prefix may be included with any of the basic job status options to force the printing of an actual account number instead of the standard output which might include "[SELF]" and/or "[OPR]".

The information listed in most of the existing sections of the SYSTAT output has been expanded. All of the new features have also been added to the ANALYS (Crash analysis program) and, where necessary, to the DISPLY base program. The extensions are:

Section	Additions
Job Status	Run-Time System under which a job is running, additional state information, and, where applicable, pseudo keyboard and controlling job number information
Disk Structure	Disk logical name or pack identification, expanded comments on state of a disk
Buffer Status	Count of system errors, job count information
Run-Time Systems	Default extension, extensive new comments

TALK

The TALK program functions in V06B-02 as it did in V06A-02.

TTYSET

The DH BURST command has been removed as it is no longer necessary.

A new command, DELIMITER, has been added to allow specification of a private delimiter for a given terminal.

The PRINT command allows setting up of start-up escape sequences for use with intelligent terminals, such as the VT52 display.

The NO ESC and ESC commands now determine translation characteristics for special characters. All these features are explained in the RSTS/E System User's Guide in the section on the TTYSET program.

UMOUNT

The UMOUNT program is accessed through the MOUNT and DISMOUNT CCL commands. The following new switches have been added in V06B-02.

Switch	Description
/LOG[ICAL]:<text>	Use <text> as the logical identifier by which to refer to the pack, rather than using the PACK ID.
/NOL[OGICAL]	Do not put any logical identification into the system for this pack.
/RO[NLY]	Mount the disk specified as a read-only disk. Must be used if the drive is physically WRITE disabled.

Any non-privileged user attempting to DISMOUNT a disk is now required to specify the PACK ID for the disk in the DISMOUNT CCL command. If a mis-match occurs, the pack will not be logically dismounted. In order to prevent a disk from being DISMOUNTed by a non-privileged user, the system manager should mount the disk using a different logical name from the PACK ID. Since the logical identification is the "Name" shown on SYSTAT printouts, the non-privileged users will be unable to find out what the true PACK ID of the disk is, thus preventing them from DISMOUNTing the disk.

UTILTY
UTILT1

To provide for access to the new features of the RSTS/E monitor, the UTILTY program's command set has been greatly extended. New features include commands to ADD and REMOVE system files (swapping files, overlay files, error message file); manipulate system-wide logical device names; manipulate Run-Time Systems, enable and disable disk caching; add, remove and list CCL commands; and enhance the system manager's control over jobs in the system.

In order to provide all of the UTILTY features while running in an 8K user space, the UTILTY program was divided into two programs, UTILTY and UTILT1.

2.8.2 RSTS/E Data Manipulation Package

The RSTS/E Data Manipulation Package contains the following:

COPY	EDIT	EDITCH	FILCOM
FLINT	PIP	PIPEXT	RUNOFF

COPY

The COPY program has been modified to allow the use of the "=", as well as the "<" to separate the input and output sides of commands.

EDIT
EDITCH

The EDIT and EDITCH programs function in V06B-02 as they did in V06A-02 except for the "hidden" PATCH option to be used for CUSP maintenance purposes.

FILCOM

The FILCOM program was modified to produce a "?" (FATAL) error prefix to the "xxx DIFFERENCES FOUND" message if <xxx> is non-zero. This permits FILCOM to be used properly in a BATCH stream, obeying the new FATAL and WARNING error level conventions.

FLINT

The FLINT (FLoppy INTerchange) program is a new program providing the user with the means to interchange IBM floppy disk data sets to and from RSTS/E data files.

FLINT allows a RSTS/E user to translate a single RSTS/E file (ASCII) to an IBM (EBCDIC) floppy disk or perform an Image mode (byte-to-byte) copy. Multi-volume output is allowed on RSTS/E to IBM transferrals. FLINT also permits single or mutli-volume translation from an IBM data set to a RSTS/E file. This feature is necessitated by the fact that IBM permits multi-volume (maximum of 99) data sets.

PIP

The PIP program (8K version) functions in V06B-02 as it did in V06A-02, except that it now accepts an = as well as a < to separate the input and output sides of a command.

PIPEXT

PIPEXT was modified to make use of the alternate I/O buffer feature now available under BASIC-PLUS. The feature is used in those cases where the input and output transfer type is the same (/BL and /CO only) and the input and output buffer lengths are equal.

RUNOFF

The RUNOFF program functions in V06B-02 as it did in V06A-02.

2.8.3 RSTS/E Error Control Package

The RSTS/E Error Control Package contains the following programs:

```
ANALYS  ANALY1
ERRINT  ERRCPY  ERRDIS  ERRDET
```

ANALYS
ANALY1

The crash dump analyzing portion of the Error Control Package is comprised of two programs in the V06B-02 release. The ANALYS program obtains all user input specifications and outputs the first part of a SYSTAT-compatible crash status report. ANALY1 (chained to by ANALYS) produces the remainder of this status report, an expanded/reformatted "Octal Dump of Status" and, optionally, a "Core Dump of the Monitor". ANALY1 also retrieves error logging information saved at the time of a crash (thereby replacing the V06A-02 ERRCRS program) and stores it in a specially formatted file, by default $ERRCRS.FIL.

The new crash analysis programs are designed to operate in one of two modes:

1. Default mode - the file [0,1]INIT.SYS is examined to determine the SIL (Save Image Library) file from which to extract pertinent monitor table information. This data is used when examining the file [0,1]CRASH.SYS (the crash dump file) to produce crash status reports and to create the $ERRCRS.FIL error logging file.

2. Non-default mode - this mode has been added to aid in the trackdown of system problems reported in SPRs. The ANALYS program will allow the user to specify the crash dump, SIL and error message files. Therefore, it is highly desirable for each installation to include machine-readable copies of the crash dump and SIL files when submitting crash dumps as part of an SPR.

The status reports produced by the analysis programs incorporate all of the new SYSTAT output formats and features with one specific change: the Who column will always explicitly list the account number for each job.

ERRINT
ERRCPY
ERRDIS
ERRDET

The Error Control Package includes a completely new release of a series of programs designed to take advantage of expanded monitor error detection/logging capabilities. The package now provides the user with the means to obtain a list of possibly bad blocks on disk media as well as more informative summary and detailed error log reports.

A list of the programs in the package is presented below, along with a brief description of each program's mode of operation and function(s):

Program	Description
ERRINT	Must be RUN to start error logging procedures during time-sharing. Its purpose is to validate or, if necessary, initialize the error logging file $ERRLOG.FIL. The program also determines if the error crash file $ERRCRS.FIL is to be appended to the error logging file.
ERRCPY	CHAINed to from ERRINT, extracts, formats and stores error messages in the error logging file $ERRLOG.FIL. If directed to by ERRINT, appends the error crash file $ERRCRS.FIL to the error logging file.
ERRDIS	This program determines user report requirements by means of an interactive dialogue. Will zero the error logging file if directed to by the user. Produces a summary report of the contents of the error logging file if one is requested.
ERRDET	CHAINed to from ERRDIS, produces the detailed report from the error logging file and will compile a bad block report if necessary.

The Error Control Package utilizes three files during its operation. $ERRLOG.FIL and $ERRCRS.FIL, created by ERRINT and ANALYS respectively, are specially formatted disk files used to store error messages. The third file, $ERRDAT.FIL, is supplied as part of the package and must be present if reports are to be generated. This data file contains specific error analysis information, such as error mnemonics, related registers and register bit descriptions.

ERRDET scans for possibly bad disk blocks by checking logged disk errors for such things as unreadable data on the last retry for a given read operation. Certain disk errors, such as those on RK06 disk drives, undergo extended checks. The program automatically checks disk errors for possibly bad blocks if the full detail report is requested; a separate listing, containing a brief description of disks with possibly bad blocks is also obtainable.

The Error Control Package provides a valuable diagnostic tool in the analysis of hardware and/or system level software problems. Each installation is encouraged to read the RSTS/E System Manager's Guide for further information on the package.

2.8.4 RSTS/E Backup Package

The RSTS/E V06B-02 Backup Package is a completely new release of a series of fifteen programs designed to provide flexibility, reliability and increased speed of backup and restore operations compared to the BACKUP program released in V06A-02.

The Backup Package utilizes a non-file structured, multi-volume format for the backup media, either magnetic tape or disk. This permits fast transfer of data with efficient utilization of the storage space on the backup media.

Several interrupt commands are available to the Backup Package user to allow interactive control of the package programs during backup and restore operations. Extensive new documentation has been provided to aid the system manager in making backup procedures timely and effective.

The Backup Package creates an extensive log of the actions taken during a backup/restore operation. This log includes complete timing and sequence flagging of all operations and shows the results of file selection and transfer, any errors which occur, and possible conflicts which render the backed up files suspect. The Backup Package takes advantage of the "read-regardless" mode of file access to obtain something from every file selected, no matter what state the file is in.

The Backup Package will also run detached under the supervision of OPSER, so that the user may still control the backup/restore operation while performing other duties.

A list of the programs in the package is presented below, along with a brief statement of each program's operation:

Program	Function
BACKUP	Command decoding and work-file initialization.
BACCON	Package operation flow controller.
BACLST	Provides a comprehensive listing of the backup/restore operation log.
BACDIR	Performs all directory look-up activities.
BACMNT	Performs all mounting/dismounting of backup volumes.
BACDSK	Creates the backup file structure on disks used for a backup volume.
BACLAB	Reads a label off a RSTS/E or BACKUP disk
BACKTO	Performs the RSTS/E device to backup volume transfer.
BACFRM	Performs the backup volume to RSTS/E device transfer.
BACCOM	Performs all transfer verification operations.
BACDMP	Dumps the backup operation index to the backup volume.
BACDEL	Performs all file deletion operations.
BACLOD	Loads a backup index file for restoration of files from a backup volume to a RSTS/E device.
BACENT	Enters accounts on the RSTS/E device.
BACPRM	Creates a prompting and special parameter table used by the BACKUP program.

The Backup Package maintains all file descriptive information, including attributes, and keeps track of bad blocks. Any new bad blocks found during backup operations are added to the destination medium bad block file (BADS.SYS) as required. The package also permits movement of files across account boundaries if needed. An extensive set of selection options is available for specifying which files, groups of files, accounts or entire devices is selected during a backup or restore operation. The user is encouraged to read the RSTS/E System Manager's Guide for further information on the Backup Package.

2.8.5 RSTS/E Spooling and Operator Services Package

The RSTS/E Spooling and Operator Services Package contains the following programs:

```
OPSER   OPSRUN
QUE     QUEMAN  QUMRUN
CHARS   SPOOL   SPLIDL  SPLRUN
BATDCD  BATCH   BATIDL  BATDEC  BATRUN
```

OPSER
OPSRUN

The Operator Services facility, comprised of the programs OPSER and OPSRUN, is a new feature appearing for the first time in V06B-02. OPSER is used to initialize the facility and OPSRUN performs command and message switching functions.

The facility (hereafter called simply OPSER) is designed to provide centralized operator message and function handling. OPSER allows system programs to communicate with a system operator instead of requiring attachment of a terminal for control functions. When directed, OPSER may permit users in non-privileged accounts access to its command features.

The Operator Services Console (OSC) provides OPSER with the ability to direct its output requirements toward selected terminals in the RSTS/E system (does not have to be KB0:).

OPSER's command repertoire provides a wide range of functions concerned with controlling OPSER's environment and/or data communicated between OPSER and cooperating jobs in the system (such as the line printer and BATCH spooling programs). The line printer and BATCH spooling programs, along with the QUE and queue manager (QUEMAN) facilities all utilize OPSER's facilities.

The PLEASE program takes advantage of the fact that OPSER is present and running in the system, and will allow users to send messages and commands to OPSER. The reader is encouraged to read the appropriate sections of the RSTS/E System Manager's Guide on the OPSER facility for further details.

QUE

The job queueing program, QUE, has undergone major modification for the V06B-02 release. Queueing is now allowed only from disk devices, and the maximum number of file specifications which can be queued in a single request has been lowered to eleven.

New options have been added to the command structure of QUE, requiring that the minimum number of characters necessary for each switch is two. New switches allow specification of number of copies of an entire job, FORM name specification for line printer spooling, queued job status, selection of jobs by sequence number, and multiple mode values.

Changes to old switch specifications include AFTER date and time and extended forms control. All file switch options can be specified as job options and thus apply to all files specified in the job request. The options may be overridden for specific files in the request by re-issuing the option value on the individual file specification.

The "Q" and "K" command switch processing has been modified. A new command, MODIFY ("M"), has been added to allow the user to change characteristics of a job which has been queued but not yet sent to a spooler.

The format of the report generated by the LIST ("L") command has been modified to allow all specified switch information to be displayed, along with the associated job and file specification data.

The CCL command QUE has been changed to the form QU[EUE] and, if entered with no arguments or switches, is equivalent to RUN $QUE. The handling of program queueing, via CHAIN entry to QUE, has been improved to provide an error text message to be returned if errors result from processing the queue request.

QUEMAN
QUMRUN

The queue manager program, QUEMAN, has undergone extensive revision. There now exist two programs: QUEMAN for initializing the queue files and restarting the queue manager functions, and QUMRUN for on-line processing of requests from the queueing and spooling programs.

The system queue file, QUEUE.SYS, has structural chacteristics different from V06A, and is now extensively checked for consistency when the QUEMAN program is run to start queueing operations. Every effort is made to ensure, if possible, that no requests entered into the queue from previous time sharing sessions are lost.

More functionality has been added to the QUEMAN and QUMRUN programs to reflect the increased capabilities of the spooling programs and to properly interface the queueing package to OPSER. Internal structure changes have been made to allow expansion of the number of on-line spooling programs (SPOOL, BATCH) to 16.

The queue manager now depends on an interface with OPSER to effect communication with an operator. The need to attach a terminal to QUEMAN for control purposes has been eliminated. A more extensive command vocabulary has been implemented for the queue manager; these commands are sent to QUEMAN and QUMRUN via OPSER's command facility. Shutting the queue manager down is also handled through OPSER under direction of the system shutdown program SHUTUP. OPSER will properly sequence the shutting down of the spooling programs and the queue manager to ensure proper retention of the queue file information.

Spooling through the queue manager can now be performed only from disk based files with the V06B-02 release. The reader is encouraged to read the RSTS/E System Manager's Guide for more information on QUEMAN and QUMRUN.

CHARS
SPOOL
SPLIDL
SPLRUN

The SPOOL program released in V06A-02 has been entirely re-written, and is now a series of three programs which perform the following functions:

- support of keyboard devices as spooled devices, including simulation, where necessary, of top-of-form operations, LPFORM handling, lower-case to upper-case conversion, and most other line printer characteristics not supported in the RSTS/E terminal handler.

- support of restart capability to the "page", file, and complete job levels, as well as a requeue facility to break a print job for later resumption of printing at the break point.

- support of special forms handling on all spooled devices.

- support of interruption commands for extensive operator control of spooling operations.

The following describe the functions performed by each of the three programs in the line printer spooling package:

SPOOL - Start up program for the package. Sets characteristics of the spooled device and initializes the internal workfile. CHAINs to the idler program SPLIDL.

SPLIDL - Idling program. Holds a dialogue with the queue manager to get new jobs. Handles all forms change operations except actual alignment. When it has received a complete job from the queue manager, or when a forms change request requires re-alignment of the form, SPLIDL CHAINs to the print program SPLRUN.

SPLRUN - Does all actual manipulation of the spooled device. Prints jobs and aligns forms. When done with a job or a forms alignment, CHAINs back to the idler program SPLIDL.

The CHARS program must be RUN (usually during installation of the programs into the system library) before any spooling operations are initiated. CHARS creates a special data file which SPLRUN uses to create the block print lettering on the job and file header pages. The CHARS.QUE file created by CHARS.BAS in V06A-02 is not usable by the V06B-02 spooling package.

The reader is encouraged to read the appropriate sections of the RSTS/E System Manager's Guide for more information on the setup and operation of the spooling package.

BATDCD
BATCH
BATIDL
BATDEC
BATRUN

The BATCH program has been entirely re-written and is now a series of four programs. The following new features have been added to the package:

- operator and queue manager communication has been re-written to interface to the revised queue manager and new OPSER facility.
- decoding and execution of the BATCH job stream commands has been divided into two distinct phases.
- both the system manager and users have been given more control over the handling of the BATCH job log.
- both the system manager and users have been given more control over job execution through detection of two levels of errors, FATAL and WARNING.
- a CPU time limit switch has been added.
- a FORTRAN processing command has been added.
- the BASIC processing command has been modified to allow use of BASIC-PLUS II.
- DENSITY/PARITY setting capabilities have been added to the MOUNT function.

The following describe the functions performed by each of the four programs in the BATCH package:

BATCH - Start up program for the package. Sets characteristics of the BATCH spooling device and initializes the internal workfile. CHAINs to the idler program BATIDL.

BATIDL - Idling program. Holds a dialogue with the queue manager to get new jobs. When it has received a complete job from the queue manager, CHAINs to the command decoding program BATDEC.

BATDEC - Command decoder. Reads the BATCH job stream file and decodes the commands and data into an internal language for presentation to the processing program BATRUN. When complete, CHAINs to the processing program.

BATRUN - Does all actual BATCH job processing. When done with a job, CHAINs back to the idling program BATIDL.

The BATDCD program must be run (usually during installation of the programs into the system library) before any BATCH processing operations are initiated. BATDCD creates a special data file which BATDEC uses in the command decode operation. The BATCH.DCD file created by BATDCD.BAS in V06A-02 is not usable by the V06B-02 BATCH package.

The reader is encouraged to read the appropriate sections of the RSTS/E System Manager's Guide for more information on the setup and operation of BATCH.

2.8.6 RSTS/E Run-Time System Support Package

The RTSLIB module functions in V06B-02 as it did in V06A-02.

2.8.7 RSTS/E User Environment Test Package

The RSTS/E V06B-02 User Environment Test Package is a completely new release designed to test RSTS/E installations from the user point of view. This package replaces the System Reliability Test programs provided with previous RSTS/E releases.

The goal of UETP is to exercise a RSTS/E system by simulating a "typical" user job mix. The package provides for user selected testing of various system components, in short single test modes or with several tests functioning in parallel over a long period of time. UETP has been used by DIGITAL manufacturing groups for some time as the final acceptance test procedure before a RSTS/E system leaves the plant.

The individual exerciser programs in UETP are also designed to be run by themselves for single device or function testing.

A list of the programs in the package and their functions is presented below:

Program	Function
UETP	Query Program
BUILDR	UETP Library Builder
ACCTST	UETP main control and query program
AC0TST	Second UETP control program
AC1TST	Third UETP control program
AC2TST	Fourth UETP control program
AC3TST	Fifth UETP control program
AC4TST	UETP control for OPTION PRE-TEST
AC5TST	UETP interactive test control program
ERRDPY	Error display header program
ERRDPA	Error display summary program
ERRDP1	Error display detail program
ERRDP2	Error display disk data program
DMEXER	RK06 disk exerciser program
DSEXER	RS03/RS04 disk exerciser program
DFEXER	RF11 disk exerciser program
DKEXER	RK05 disk exerciser program
DXEXER	RX01 floppy disk exerciser program
MTEXER	TM02/TU10/TU16/TU45 Magtape exerciser program
DTEXER	DECtape exerciser program
CPEXER	Central Processing Unit compute bound tester program
CREXER	CR11/CD11 card reader test program
PPEXER	Paper tape reader test program
PREXER	Paper tape punch test program
KBEXER	Individual keyboard test program
SCRIPT	Reads SCRIPT files into interactive code
SCRPTB	Runs several sub-jobs via pseudo keyboards
CVTSCP	Converts SCRIPT files to user job files
OPTPRM	Non-file structured RK06 mechanical test
OPTPRK	Non-file structured RK05 mechanical test
OPTPRB	Non-file structured RP04/RP05/RP06 mechanical test

TDLSRT	Disk sort test main program
TDLXQK	Disk sort subcontrol program
TDLMQK	Disk sort subcontrol program
TDLOQK	Disk sort subcontrol program
TDLSQK	Disk sort subcontrol program
CRFILE	Creates a tape-sort random data file
TAPSRT	Magtape sort program
TAPSRU	Magtape sort program
DECMRG	Disk/tape merge program
NEWRB	RP04/RP05/RP06 disk exerciser program
NEWRP	RP02/RP03 disk exerciser program
VERIFY	Verifies result of the tape sort test
RANMAK	Random access file creator program
JSTATS	Job statistics program
DSTATS	Disk statistics program
CPUTST	CPU compute bound test program
FILMIN	Disk file manipulation using virtual arrays
UDA	Creates files used by TDLSRT
FLOPPY	Floppy disk sort/merge program
DA	Creates files used by TDLSRT
PKBSNE	Special pseudo keyboard test program

The following data files are used:

CLUMSY.SCP	FILES .SCP	IMMED .SCP	EDIT .SCP
TTY .SCP	RANDAC.SCP	RELEAS.SEQ	RELEAS.SRT
CPU .SCP	VIRSTR.SCP	MTSORT.SCP	

2.9 Documentation

Documentation Directory

The Documentation Directory was updated to reflect the changes and new additions to the documentation available for the V06B-02 release of RSTS/E.

RSTS/E System Generation Manual

The RSTS/E System Generation Manual was reorganized and re-written for this release to include the new system generation procedures and to simplify system generation for the experienced user.

A new feature of the manual is the system generation "cookbook." The cookbook, which is printed on colored paper, contains a summary of the procedures used in system generation. Intended for users experienced at generating RSTS/E V06B-02, the cookbook includes summaries of the initialization options, configuration questions, and system library BUILD procedures in a step-by-step format.

Another addition to the manual is information on regenerating RSTS/E monitors and BASIC-PLUS Run-Time Systems on-line during time sharing (a new feature for this release).

The optional language and auxiliary Run-Time System BUILD procedures were removed from the manual. This information will appear as articles in the Release Notes and as documentation supplied with the optional feature installation kit.

RSTS/E System Manager's Guide

The RSTS/E System Manager's Guide has been updated to describe added functionality. Chapter 3 contains new INIT and SHUTUP procedures, which have changed significantly. Chapter 5 has been completely re-written to describe the Spooling and Operator Services Package. Chapter 6 has been re-written to reflect the new monitor error logging facility and related programs.

Many of the sections of Chapter 7 have been updated and re-written. New UTILTY command descriptions reflect system program implementation of new SYS function calls. The section on Run-Time Systems was also re-written. The sections on DSKINT and REORDR were revised to reflect new versions of the programs.

Chapter 8 has been added to the manual to present the privilege features available in the Backup Package.

RSTS/E System User's Guide

The RSTS/E System User's Guide has been expanded and editorially revised for improved readability and ease of reference. The manual is organized in five parts. Within each part, commands and/or programs are organized according to function, and each part contains a brief introductory chapter.

Part I is intended for new RSTS/E users who may have only slight knowledge of computers and the concept of time-sharing: it provides an overview of the RSTS/E system and its resources.

Part II describes the system commands that manipulate devices and defaults (ASSIGN, DEASSIGN and REASSIGN commands dominate this part).

Part III describes the BASIC-PLUS system commands, which enable the user to write, run, edit, modify and debug BASIC-PLUS programs.

Part IV describes the system library programs, including the new Backup Package and the FLINT (Floppy Disk Interchange) program.

Part V contains descriptions of peripheral hardware available for a PDP-11 installation running RSTS/E.

BASIC-PLUS Programming Manual

This edition of the RSTS/E BASIC-PLUS Language Reference Manual documents language enhancements incorporated in RSTS/E BASIC-PLUS V06B. These enhancements include:

1. String arithmetic functions (SUM$, DIF$, PROD$, QUO$, PLACE$, and COMP%), which perform arithmetic or comparison operations on numeric string arguments up to 60 characters long, allowing extended arithmetic precision. Another function, NUM1$, converts integer and floating point values into numeric strings.

2. The EXTEND mode option, which allows 30-character variable names, but with strict format requirements.

3. Unlimited source line length.

4. Provision for alternate I/O buffers on GET and PUT operations; this allows GET and PUT statements to access the same channel buffer, thus eliminating intermediate data movement between channel buffers.

This edition corrects a number of errors and clarifies certain potentially misleading information from the previous edition. Some material that, in DIGITAL's judgement, is not properly within the scope of a Language Manual has been deleted. The manual uses examples that reflect good programming practice. The overall organization of the previous edition is largely retained.

RSTS/E Programming Manual

The RSTS/E Programming Manual has been updated to reflect new features and improve usability. Chapter 1 contains a new description of privilege which explains temporary privilege and the ability to drop and regain temporary privilege. The disk description includes all new MODE values and new file update capabilities. A section on the Null Device (NL:) was also added to Chapter 1.

The ANSI magnetic tape dicussion in Chapter 2 was updated to clarify the usage of CLUSTERSIZE and FILESIZE options in the OPEN FOR OUTPUT statement.

Chapter 3 discusses new MODE and RECORD values which allow more programmed control of line printer output.

Chapter 4 has been reorganized slightly and a discussion of new MODE values for terminal input and output has been added.

Chapter 7 contains updated and added descriptions of system function SYS calls.

The following new features have been added in the FIP SYS call section:

1. adding and deleting CCL commands
2. adding, removing and changing system logical names
3. reading and writing file attributes
4. adding and removing system files

The following FIP SYS call descriptions have been updated to reflect added functionality.

1. The drop temporary privileges call now allows a program to drop and regain privileges.
2. The Run-Time System call now has added functions for handling Run-Time Systems.
3. The Get Monitor Tables calls now provide additional information.
4. The filename string scan calls now handle monitor level switches.
5. The disk pack service call allows a program to specify a logical pack name and set special mount characteristics.
6. The Set Terminal Characteristics call has additional features.
7. The Detach call can detach another job and can close all channels on which the job console terminal is OPEN.
8. The Return Error Message call returns an error severity indicator ("?" or "%") as part of the error message text string.

Other FIP SYS call descriptions have been changed slightly to correct errors and improve understandability.

The expanded functionality of inter-job communication is described in a chapter of its own, Chapter 8.

Chapter 9 contains a revised description of CCL processing and a new explanation of the CVT functions for converting integer and floating-point data. Also included are new features for handling logged out commands and information on LOGIN's setting of the SWAP MAX value for each new job to 28K words.

Appendix C describes the system error procedures and contains new information on error severity indicators.

RSTS/E RUNOFF User's Manual

No changes were made to the RSTS/E RUNOFF User's Manual.

RSTS/E Text Editor Manual

No changes were made to the RSTS/E Text Editor Manual.

RSTS/E V06B-02 Release Notes

The Release Notes have been re-written and re-organized for the V06B-02 release.

RSTS/E Pocket Guide

The RSTS/E Pocket Guide provides fast reference to the RSTS/E system. Intended for users who know the system but require a reference aid, the RSTS/E Pocket Guide includes summaries of the following:

- system standards and file information
- system resource commands
- library programs that run by CCL commands
- the new Backup Package features
- BASIC-PLUS system commands
- BASIC-PLUS language

Typography and format have been designed for ease of reference and visual appeal.

2.10 Old Problems Corrected

Various system problems which were found and fixed after the release of RSTS/E Version V06A-02 are listed below. For convenience, the problems are grouped according to the area in which they occurred.

2.10.1 RSTS/E Monitor

DH11 output problems are fixed.

All RP04 error conditions are now handled correctly.

The "Read Accounting Data" SYS call now reports 65535 blocks in use if the account has more than 65535 blocks allocated to it.

Terminals OPEN in binary input mode are now properly closed when a job is KILLed.

The "Reattach" SYS call can now be issued with OPEN channels.

Executable files (protection code >=64) have an altered protection code scheme and can be run even if OPEN in update mode.

2.10.2 BASIC-PLUS Language and Run-Time System

A failure of MAT INV function when the SCALE factor was 6 was fixed.

BASIC-PLUS now handles an immediate mode RETURN command properly.

A CLOSE followed by an INPUT LINE statement will now return an end-of-file indication.

Several problems concerning array and I/O buffer allocations causing errors were fixed.

A <cr> appearing alone in a string of characters read with an INPUT or INPUT LINE command no longer disappears unless it is followed immediately by a NULL (CHR$(0%)) character.

A problem with negative values in FIELD statements was fixed.

Several cases where subscripting errors were not reported properly were corrected.

A problem with re-dimensioning of arrays causing memory management errors was fixed.

Several floating-point to ASCII conversion errors with "funny" numbers were remedied.

A problem with the statement SPACE$(32767%+1%) was fixed.

The construct FOR I% = 1% WHILE 1% now works properly.

A problem with zero length strings in virtual arrays was fixed.

2.10.3 Commonly Used System Programs (CUSPs)

BATCH now correctly handles the decimal arithmetic math package.

DIRECT now reports accounts correctly when wildcards and multiple public disks are used.

EDIT did not execute a "0J" command identically to a "0A" command, as the documentation states it should; the "P" command did not work properly.

The RUN BURST is now properly setup by LOGIN for each new job.

PIP did not give the correct error message when an attempt was made to zero a write protected magtape.

PIPEXT did not report an error when a wildcard PPN was specified for a disk device; it reported a misleading error message when a wildcard file specification was given for the input file and the output device was non-file structured; under certain circumstances, it wrote meaningless data to magtape from a one record file when the /FORMAT:V switch was used; it reported a CLUSTERSIZE of 0 instead of 256. All of these problems are fixed in V06B.

QUEMAN problems in handling and maintaining the queue file have been corrected.

Numerous problems in RUNOFF were corrected.

SYSTAT now prevents non-privileged users from writing files across accounts.

2.10.4 Run-Time Systems and Support

The auxiliary Run-Time System package has been augmented to include utilities and libraries necessary for the installation and maintenance of new language products supported by RSTS/E V06B, such as BASIC-PLUS II and new versions of COBOL and RPG-II. The following files are included:

RSX.RTS	The Run-Time System used by the following three utilities.
TKB.TSK	Task Builder to be used for linking BASIC-PLUS II subprograms.
MAC.TSK	MACRO-11 assembler to be used either as a software maintenance tool by DIGITAL or for the purpose of generating object files from MACRO-11 language source files produced by the BASIC-PLUS II compiler. Any other use of this utility is not supported by DIGITAL.
LBR.TSK	Librarian utility to be used only as a software maintenance tool by DIGITAL.

SYSLIB.OLB	System object library, to be used only as a software maintenance tool by DIGITAL.
RSXMAC.SML	System Macro library, to be used only as a software maintenance tool by DIGITAL.
FLX.BAC	A file translation program for printing .MAP or listing files produced by the three .TSK utilities described above when the Spooling Package is not available. Alternatively may be used for converting object files and others between DOS Formatted Binary and RSX formats.

2.11 Upgrading from V06A-02 to V06B-02

When you are upgrading from RSTS/E V06A (or any earlier version of RSTS/E) to RSTS/E V06B-02, you may place the V06B system on your V06A system disk, or on any other disk which contains a valid RSTS/E file structure. This disk must contain enough contiguous space to accommodate the new system.

You may place either your SYSGEN system or your target system on any RSTS/E file structured disk. Before doing so, be sure to create a backup copy of the disk. Once you start the initialization code COPY procedure, your disk is no longer usable as a V06A system disk. After backing up the disk, you can create the V06B system on it, following the guidelines in this section. First, create as much contiguous space as possible, as described in the next two sections.

2.11.1 Creating a V06B-02 System on a V06A System Disk

If you are using a V06A system disk as your V06B system disk, you must remove any monitor files used by V06A. First, delete any .RTS files in account [0,1]. These files will not be usable under V06B. Next, shut down your system and use the REFRESH option of V06A INIT. Using REFRESH, change the size of RSTS.CIL to the minimum allowable (the REQUIRED SIZE listed in the table). Change the size of SWAP0.SYS to the minimum allowable, namely 32 blocks. Move SWAP0.SYS to the system disk (if it is not already there), and answer the BASE question with 0 (not LINE FEED) to move it to the lowest possible position on the disk. Delete SWAP1.SYS, SWAP2.SYS, and SWAP3.SYS. Move OVR.SYS and ERR.SYS to the CIL. Delete BUFF.SYS. Answer NO to the CRASH query.

The V06A system disk should now contain a large contiguous area. Bootstrap your V06B medium (distribution or intermediate), and use the COPY option to copy system files to the target disk. Even if this fails due to lack of contiguous space on the disk, COPY may have copied at least INIT.SYS. In any event, bootstrap the target disk. If the V06A prompt appears, there is no room for INIT.SYS on the target disk, and you must use a new disk for your V06B system. If the V06B prompt appears, the RSTS/E V06B initialization code (and possibly more or all of the V06B system files) has been copied. Use the REFRESH option of V06B INIT to clean and refresh the new system disk. With the FILE suboption, delete RSTS.CIL and SWAP0.SYS. This extends the contiguous area on disk. Use the BOOT option to reboot your V06B medium (distribution or intermediate), and again use the COPY option to copy system files to the target disk. Specify 'D' (delete) to delete and recopy any V06B files that were copied the first time. If the COPY option fails, you must use a new disk for your V06B system. If it succeeds, you may proceed to tailor your RSTS/E V06B system.

2.11.2 Creating a V06B-02 System on a V06A Nonsystem Disk

If you are using a nonsystem V06A disk as your V06B system disk, you do not have any monitor files to eliminate. Use the COPY option of V06B INIT to try to create your system on this disk. If COPY fails due to lack of contiguous space (i.e. produces the NO ROOM FOR FILE ON OUTPUT DISK error message), try to create more space. If you have not already done so, bring up your V06A system and use BACKUP to create a copy of all files and accounts on the disk. Shut down your system and use V06B DSKINT and COPY to build your V06B system on the nonsystem V06A disk. Then bring up your V06A system again and use BACKUP to restore all files to the disk. You may specify restoration to the public structure in order to distribute the files over all public packs. Shut down V06A and use the V06A BOOT option to bootstrap V06B INIT from your new system disk. Use the REFRESH option of V06B INIT to refresh the old V06A system disk, and use the FILE suboption to delete system files from that disk (RSTS.CIL, SWAPn.SYS, OVR.SYS, ERR.SYS, BUFF.SYS, and CRASH.SYS). You may now continue tailoring your V06B system.

3.0 Contents of RSTS/E V06B-02 Distribution Kits

For each media element, the catalogue number and a description are given. Where applicable, the logical disk name is also given.

MONITOR

INIT/MONITOR/BASIC-PLUS (Combined)

Label	Logical Name	Description
DEC-11-ORSPA-E-MC7		TU10 7-track Magnetic Tape
DEC-11-ORSPA-E-MC9		TU10/TS03/TU16/TU45 9-track Magnetic Tape
DEC-11-ORSPA-E-HC	ORSPAE	RK05 DECpack
DEC-11-ORSPA-E-BC	ORSPAE	RK06 DEC Data Cartridge

INIT/MONITOR/BASIC-PLUS (DECtape)

DEC-11-ORSBA-E-UC1	DECtape (INIT, SYSGEN Monitor Etc.)
DEC-11-ORSBA-E-UC2	DECtape (SYSGEN CUSPs)
DEC-11-ORSEA-E-UC1	DECtape (MONITOR .MAC Files)
DEC-11-ORSEA-E-UC2	DECtape (MONITOR .OBJ Files)
DEC-11-ORSEA-E-UC3	DECtape (BASIC-PLUS Files)

UTILITIES

UTILITY PROGRAMS (Combined)

DEC-11-ORSLA-E-BA	ORSLAE	RK06 DEC Data Cartridge

UTILITY PROGRAMS (Small programs)

DEC-11-ORS1A-E-MA7		TU10 7-track Magnetic Tape
DEC-11-ORS1A-E-MA9		TU10/TS03/TU16/TU45 9-track Magnetic Tape
DEC-11-ORSLA-E-HA1	ORSLAE	RK05 DECpack

UTILITY PROGRAMS (Large programs)

DEC-11-ORS2A-E-MA7		TU10 7-track Magnetic Tape
DEC-11-ORS2A-E-MA9		TU10/TS03/TU16/TU45 9-track Magnetic Tape
DEC-11-ORSLA-E-HA2	ORSLAE	RK05 DECpack

UTILITY PROGRAMS (DECtape)

DEC-11-ORSCA-E-UA1	Library (Small programs 1)
DEC-11-ORSCA-E-UA2	Library (Small programs 2)
DEC-11-ORSCA-E-UA3	Library (Large programs)
DEC-11-ORSCA-E-UA4	Spooling Package 1
DEC-11-ORSCA-E-UA5	Spooling Package 2
DEC-11-ORSCA-E-UA6	Spooling Package 3
DEC-11-ORSCA-E-UA7	BACKUP Package 1
DEC-11-ORSCA-E-UA8	BACKUP Package 2
DEC-11-ORSCA-E-UA9	BACKUP Package 3
DEC-11-ORSCA-E-UA10	BACKUP Package 4
DEC-11-ORSCA-E-UA11	Special Programs 1
DEC-11-ORSCA-E-UA12	Special Programs 2
DEC-11-ORSCA-E-UA13	UETP Programs 1
DEC-11-ORSCA-E-UA14	UETP Programs 2

4.0 Notebook Maintenance

Each month the Software Dispatch should be taken apart and the new articles inserted in order of sequence. If an article is a replacement, discard the superseded article. An article addresses a single topic and its title gives the symptom or a one-line problem description. This information appears at the top of each page:

RSTS/E Software Dispatch, (date)

RSTS/E V06B-02 Seq 10.21.1 O
System Utilities Package
SHUTUP Page 1 of 2

The System COMPONENTS.SUBCOMPONENTS are listed in Section 4.1. Articles for each subcomponent are assigned a sequence number which uniquely identifies the article within that component category. As each new article is published for a sub-component, it is assigned the next higher sequence number. The sequence numbers are defined as COMPONENT.SUBCOMPONENT.ISSUE number. An example would be 6.2.11. Replacement articles will be noted as such.

Articles for subordinate software that is not unique to RSTS/E are to be filed by product and then component. Articles are sequenced by component after this sorting process. In some cases, articles are so few that they are sequenced only by product. Missing sequence numbers indicate problems unique to other operating systems. The cumulative index is published in the Dispatch to help the user maintain the notebook for the complete RSTS/E system.

A flag may appear after the sequence number:

M = Mandatory patch
O = Optional patch
R = Restriction
N = Note

4.1 RSTS/E COMPONENT.SUB-COMPONENT Assignments

The assignments outlined below appear in the COMPONENT.SUB-COMPONENT part of the SEQUENCE NUMBER, which appears on articles to be filed in Section 7.0 of this notebook.

RSTS/E MONITOR

1.0 Initialization
- 1.1 INIT.SYS Program Patches
- 1.2 INIT.SYS Program Notes

2.0 System Generation
- 2.1 SYSGEN Patches
- 2.2 SYSGEN Notes

3.0 Executive
- 3.1 Monitor Patches
- 3.2 Monitor Notes
- 3.3 Terminal Service Patches
- 3.4 Terminal Service Notes
- 3.5 File Processor Patches
- 3.6 File Processor Notes
- 3.7 Device Driver Patches
- 3.8 Device Driver Notes

4.0 BASIC-PLUS
- 4.1 BASIC-PLUS Patches
- 4.2 BASIC-PLUS Notes
- 4.3 Programming Hints
- 4.4 Math Packages
- 4.5 Matrix Functions
- 4.6 PRINT-USING
- 4.7 Record I/O
- 4.8 Virtual Core
- 4.9 String Arithmetic

5.0 RSTS/E 2780 Device Driver
- 5.1 2780 Device Driver Patches
- 5.2 2780 Device Driver Notes

6.0 (Reserved)

7.0 (Reserved)

8.0 (Reserved)

9.0 (Reserved)

RSTS/E UTILITIES

10.0 System Utilities Package
- 10.1 Package Notes
- 10.2 BUILD Utility
- 10.3 DIRECT
- 10.4 DISPLY
- 10.5 VT05 Adaptation of DISPLY
- 10.6 VT50 Adaptation of DISPLY
- 10.7 (Reserved)
- 10.8 DSKINT
- 10.9 GRIPE
- 10.10 INIT
- 10.11 INUSE
- 10.12 LOGIN
- 10.13 LOGOUT
- 10.14 MONEY
- 10.15 ODT
- 10.16 PLEASE
- 10.17 PRIOR
- 10.18 QUOLST
- 10.19 REACT
- 10.20 REORDR
- 10.21 SHUTUP
- 10.22 SYSCAT
- 10.23 SYSTAT
- 10.24 TALK
- 10.25 TTYSET
- 10.26 UMOUNT
- 10.27 UTILTY
- 10.28 UTILT1

11.0 Data Manipulation Package
- 11.1 Package Notes
- 11.2 COPY
- 11.3 EDIT
- 11.4 EDITCH
- 11.5 FILCOM
- 11.6 FLINT
- 11.7 PIP
- 11.8 PIPEXT
- 11.9 RUNOFF

12.0 Error Control Package
- 12.1 Package Notes
- 12.2 ANALY1
- 12.3 ANALYS
- 12.4 ERRCPY
- 12.5 ERRDET
- 12.6 ERRDIS
- 12.7 ERRINT

13.0 Backup Package
- 13.1 Package Notes
- 13.2 BACCOM
- 13.3 BACCON
- 13.4 BACDEL
- 13.5 BACDIR
- 13.6 BACDMP
- 13.7 BACDSK
- 13.8 BACENT
- 13.9 BACFRM
- 13.10 BACLAB
- 13.11 BACLOD
- 13.12 BACLST
- 13.13 BACMNT
- 13.14 BACPRM
- 13.15 BACKTO
- 13.16 BACKUP

14.0 Spooling and Operator Services Package
- 14.1 Package Notes
- 14.2 OPSER
- 14.3 OPSRUN
- 14.4 QUE
- 14.5 QUEMAN
- 14.6 QUMRUN
- 14.7 CHARS
- 14.8 SPOOL
- 14.9 SPLIDL
- 14.10 SPLRUN
- 14.11 BATDCD
- 14.12 BATCH
- 14.13 BATIDL
- 14.14 BATDEC
- 14.15 BATRUN

15.0 User Environment Test Package
- 15.1 Package Notes
- 15.2 UETP
- 15.3 ACCTST
- 15.4 AC0TST
- 15.5 AC1TST
- 15.6 AC2TST
- 15.7 AC3TST
- 15.8 AC4TST
- 15.9 AC5TST
- 15.10 ERRDPY
- 15.11 ERRDPA
- 15.12 ERRDP1
- 15.13 ERRDP2
- 15.14 DMEXER
- 15.15 DSEXER
- 15.16 DFEXER
- 15.17 DKEXER
- 15.18 DXEXER
- 15.19 MTEXER
- 15.20 DTEXER
- 15.21 CPEXER
- 15.22 CREXER

15.23 PPEXER
15.24 PREXER
15.25 KBEXER
15.26 SCRIPT
15.27 SCRPTB
15.28 CVTSCP
15.29 OPTPRM
15.30 OPTPRK
15.31 OPTPRB
15.32 TDLSRT
15.33 TDLXQK
15.34 TDLMQK
15.35 TDLOQK
15.36 TDLSQK
15.37 CRFILE
15.38 TAPSRT
15.39 TAPSRU
15.40 DECMRG
15.41 NEWRB
15.42 NEWRP
15.43 VERIFY
15.44 RANMAK
15.45 JSTATS
15.46 DSTATS
15.47 CPUTST
15.48 FILMIN
15.49 UDA
15.50 FLOPPY
15.51 DA
15.52 BUILDR
15.53 PKBSNE

16.0 System Management Guidelines
16.1 Recovery of Disk Data
16.2 Operator Services
16.3 System Backup

17.0 (Reserved)

18.0 (Reserved)

RSTS/E DOCUMENTATION

19.0	Documentation	
19.1	Documentation Directory	DEC-11-ORDAA-B-D
19.2	RSTS/E System Generation Manual	DEC-11-ORGNA-B-D
19.3	RSTS/E System Manager's Guide	DEC-11-ORSMD-B-D
19.4	RSTS/E System User's Guide	DEC-11-ORSUB-A-D
19.5	BASIC-PLUS Language Manual	DEC-11-ORBPB-E-D
19.6	RSTS/E Programming Manual	DEC-11-ORSRB-B-D
19.7	RSTS/E RUNOFF User's Guide	DEC-11-URUNA-B-D
19.8	RSTS/E Text Editor Manual	DEC-11-UTEMA-A-D
19.9	RSTS/E V06B-02 Release Notes	DEC-11-ORINC-A-D
19.10	RSTS/E Pocket Guide	DEC-11-ORFCA-A-D

RUN-TIME SYSTEMS SUPPORT

20.0 Run-Time System Support Package
- 20.1 Package Notes
- 20.2 LNGBLD
- 20.3 TKB
- 20.4 RTSLIB.RTS
- 20.5 BIGTKB.TSK

21.0 RSX Emulator Package
- 21.1 Package Notes
- 21.2 FLX.BAC
- 21.3 RSX.RTS
- 21.4 TKB.TSK
- 21.5 MAC.TSK
- 21.6 LBR.TSK
- 21.7 SYSLIB.OLB
- 21.8 RSXMAC.SML

22.0 RT-11 Emulator Package
- 22.1 Package Notes
- 22.2 RT11.RTS
- 22.3 CREF.SAV
- 22.4 EDIT.SAV
- 22.5 HOOK.SAV
- 22.6 LIBR.SAV
- 22.7 LINK.SAV
- 22.8 LOGIN.SAV
- 22.9 LOGOUT.SAV
- 22.10 MACRO.SAV
- 22.11 PATCH.SAV
- 22.12 PIP.SAV
- 22.13 SILUS.SAV
- 22.14 SYSBAT.SAV
- 22.15 SYSGEN.SAV
- 22.16 UTILTY.SAV

4.2 Subordinate Software

23.0 FORTRAN IV/RSTS-E V01C

- 23.1 PACKAGE NOTES
- 23.2 COMPILER
- 23.3 OTS
- 23.4 RUNTIME SYSTEM

24.0 DMS-500 V02

- 24.1 PACKAGE NOTES
- 24.2 DMSUTL
- 24.3 DMSFSS
- 24.4 DMSISR
- 24.5 DMSISC
- 24.6 DMSRAR
- 24.7 DMSISA
- 24.8 DMSRAA
- 24.9 DMSFSU
- 24.10 DMSISO
- 24.11 DMSRAO
- 24.12 DMSFSX
- 24.13 IAMGEN
- 24.14 IAMCRI
- 24.15 IAMDMP
- 24.16 IAMVFY
- 24.17 IAMPRI
- 24.18 IAMFNS
- 24.19 DSORT
- 24.20 DSORTD
- 24.21 DSORTF
- 24.22 DSORTX
- 24.23 DSORTS
- 24.24 DSORTM
- 24.25 DSORTO
- 24.26 DMS-500 User's Guide

25.0 RSTS/E 2780 PACKAGE

COBOL-11 V2

SORT-11 V1

RPGII V7.06

COMPILER	V7.06
OTS	V7.06
RPGASN	V1-06
RPGDCV	V1-14
RPGDMP	V1-04
RPGESP	V1-16
RPGFIL	V2-04
RPGFNS	V1-12
RPGPCV	V1-14
RPGXEQ	V1-02

DIGITGRAPH
P. O. Box 4249
Tucson, Arizona 85717

5.0 Software Performance Reports

Each new installation is provided with Software Performance Report (SPR) forms. The SPR form enables the user to suggest enhancements to or report problems with DIGITAL software or documentation. When a problem is encountered, an SPR should be completed and mailed to the local SPR Center (see inside of back cover).

Responses will be sent to the name and address appearing on the form. Additional SPR forms may be obtained by writing to the local SPR Center. SPR response is free for a period of one year after installation and by subscription thereafter.

5.1 Software Performance Report Guidelines

This guideline for SPR completion is provided to ensure that adequate information is included to prevent delays in processing.

Every SPR form should be completely filled out. A clear description of the problem is very important and will speed response.

It is important to know the complete hardware configuration for the installation, including CPU type, system disk, amount and type of memory, hardware options (such as hardware floating-point processor), and peripherals on the system. Similarly, it is important to know what RSTS/E monitor options were specified at system generation time, including which math package and BASIC-PLUS options were selected. The easiest way to supply this information is to attach a copy of the listing of the CONFIG.MAC and SYSGEN.CTL files produced during the system generation process. Alternatively, a copy of the console terminal dialogue could be provided to show which options and procedures were selected.

In addition to the above points, it is important to specify the following items when submitting an SPR concerning a DIGITAL supplied BASIC-PLUS program:

1. The name of the CUSP.

2. The version and edit level of the CUSP. This is generally found at statement 1010 in the program listing. The optional patches included in the CUSP, if any, should also be listed.

3. Under what account(s) the CUSP failed; that is, distinguish between privileged and non-privileged behavior.

4. The PRIORITY and SWAP MAX which the CUSP was running under when it failed.

5. A terminal printout of the relevant command strings and/or input data.

6. A complete list of any modifications which have been made to the CUSP at the user's installation. (This will not only help to determine the possible cause of failure but will also suggest changes to improve CUSP performance and reliability.)

7. A listing of the contents of log files, where applicable.

Software Support is unable to analyze crash dumps submitted with SPRs unless, as an absolute minimum, listings of the system load maps are included. Submitted crash dumps should include the full "CORE DUMP OF MONITOR" output of the crash dump analysis program. It is very desirable to include, in machine readable form, the [0,1]CRASH.SYS file along with the installed monitor .SIL file (usually [0,1]RSTS.SIL). When saving these files, be sure to use the /CO switch in the PIP program command.

Before submitting an SPR, the user should review the relevant section(s) of these Release Notes to ensure that the problem has not already been solved and a solution published.

6.0 Patching Procedures

This section briefly describes the patching procedures to be used for maintenance of RSTS/E Version V06B-02. Two methods will be used for RSTS/E software: 1) Monitor patches and 2) CUSP patches. Other optional software supplied to RSTS/E installations may require different patching procedures from those about to be discussed; these alternative methods will be explained further under the associated COMPONENT.SUB-COMPONENT section for the software concerned. Unless otherwise stated, all patches published in the Release Notes and the Software Dispatch are MANDATORY patches which should be made to the affected component as soon as possible. Since not all components will be included at each installation, mandatory patches for unused components do not have to be made. The system manager, however, should file all articles, even for unused components, since the installation might choose at some future time to install additional components and those would then have to be brought up to the latest support level.

Some articles will detail the procedure to patch a component in order for the installation to use an optional feature of that component. These "Optional" patches are not required to keep a component at its current support level; however, if problems occur with a component containing optional features, the installation should be sure to report which optional features are in use when filing the Software Performance Report (SPR).

Next, some discussion of documentation conventions needs to be presented. Below is a table of syntactic elements which provides a description of certain terminal keys which are non-printing functions:

^C or CTRL/C	Control-C
<tab>	Horizontal tab
<lf>	Line feed
<ff>	Form feed
<cr>	Carriage return
^Z or CTRL/Z	Control-Z
<esc>	Escape (Alt Mode or SEL)

When one of these syntactic elements appears in the documentation, it signifies that the associated key(s) is (are) pressed if input is desired or the appropriate terminal action occurs if an output operation is being described.

At various places in the documentation, a need will arise for stating that a particular item is optional. This optionality will be indicated by enclosing the item text between brackets. Note that nesting may take place as indicated by the following example:

RUN <program name>[/DET[ACH]]

The modifier /DETACH is optional, as are the last three letters of DETACH. In other words, /DET, /DETA, /DETAC, or /DETACH is acceptable, if this modifier is used.

Note that whenever a project-programmer number is called for in the text, the [and] which may appear are explicitly stated and do not signify that the enclosed quantity is optional.

6.1 Monitor Patching

Changes to the RSTS/E monitor will be made via the PATCH option of INIT.SYS as described in the RSTS/E System Managers Guide. Section 3.4.1 describes in detail the use of this option to make changes to monitor code.

Release Note articles discussing various monitor options and problem corrections will detail the exact operator action required when using the PATCH option. The expected dialogue will be shown in the article as it should appear when the patch is made. If any deviation occurs, the process should be halted, any changes already made returned to their former values via the PATCH option, and a determination made as to where the discrepancy occurred.

6.2 Commonly Used System Program (CUSP) Patching

The RSTS/E EDIT program will serve as the CUSP program patching tool for maintenance purposes. A special patching mode has been added to EDIT (see Article 11.3.1 in Section 7.0) to facilitate EDIT's use in modifying BASIC-PLUS source files (.BAS).

Refer to the RSTS/E Text Editor Manual for complete information on the commands and capabilities of EDIT.

Each patch notice will be self-contained as a convenience to the person who will be making the change. It is recommended that the date of the change and the name of the person making the patch be written on the article at the time the program modification is performed in order to keep an up-to-date record of the state of the system library software.

7.0 Release Notes and Software Dispatch Articles

Section 7.0 is the repository for all Release Notes and articles directed at specific components of the RSTS/E system. This section will include problem solutions, optional feature "patches", documentation errata, programming hints, and tips for better operation of RSTS/E.

Each person who will generate a RSTS/E system (whether on-line under time sharing or as the initial version of the system) should become familiar with these articles. This should be a continuing process of education as sub-sections are added and updated.

Beginning on the next page is an index to all Release Notes Articles. This index will be continually updated as new articles are issued.

7.1 Cumulative Index of all Articles

Component	Sequence	
RSTS/E V06B-02		
RSTS/E INIT.SYS PROGRAM		
INIT.SYS PROGRAM PATCHES		
DC11'S AT PR5 ARE DISABLED	1.1.1 M	RN
DZ11'S ARE DISABLED	1.1.2 M	RN
CRASH DUMP/AUTO-RESTART FAILS	1.1.3 M	RN
REFRESH OPTION FAILS TO DELETE FILES	1.1.4 M	RN
UNIBUS WINDOW ALLOCATION	1.1.5 M	RN
DEVICE CHECKING TIMEOUTS	1.1.6 M	RN
INIT.SYS PROGRAM NOTES		
CLEANING DISKS AFTER SYSTEM CRASHES	1.2.1 N	RN
ADDING BAD BLOCKS TO THE BAD BLOCK FILE	1.2.2 N	RN
RP04/RP05/RP06 BOOTSTRAP	1.2.3 R	RN
RSTS/E EXECUTIVE		
MONITOR PATCHES		
MAKING RECEIVER DECLARATION NON-PRIVILEGED	3.1.1 O	RN
CHANGING POWER FAIL DELAY FACTOR	3.1.2 O	RN
MESSAGE SEND/RECEIVE BUFFER HANDLING	3.1.3 M	RN
NETWORK MESSAGE SEND FAILURE	3.1.4 M	RN
TERMINAL SERVICE PATCHES		
RESTRICTING TERMINAL MODES	3.3.1 O	RN
IMMEDIATE MODEM HANGUP	3.3.2 O	RN
CHANGING SPECIAL LOGIN TERMINAL	3.3.3 O	RN
CHANGING A TERMINAL'S INPUT BUFFER QUOTA	3.3.4 O	RN
INPUT ESCAPE SEQUENCE HANDLING	3.3.5 O	RN
JOB SET-UP FAILURE IF NO SMALL BUFFERS	3.3.6 M	RN
CONTROL/B IN ECHO CONTROL MODE	3.3.7 M	RN
LEAVING ECHO CONTROL MODE WITH PENDING INPUT	3.3.8 M	RN
TERMINAL SERVICE NOTES		
2741 HARDWARE OPTIONS REQUIRED BY RSTS/E	3.4.1 N	RN
2741 TERMINAL INTERFACE WIRING	3.4.2 N	RN
UNDERSTANDING SYNCHRONIZATION DELAYS	3.4.3 N	RN
FILE PROCESSOR PATCHES		
AUXILIARY LIBRARY ACCOUNTS	3.5.1 O	RN
CHANGING DEFAULT PROTECTION CODE	3.5.2 O	RN
RESTRICTING DISK MODES	3.5.3 O	RN
INVERTING DIRECTORY ORDERING	3.5.4 O	RN
MAKING FIP SYS CALLS PRIVILEGED	3.5.5 O	RN
DEVICE DRIVER PATCHES		
SPECIAL LINE PRINTER DRUMS	3.7.1 O	RN
CHANGING DEFAULT PRINTER FORM LENGTH	3.7.2 O	RN
TU16/TU45 MAGTAPE DENSITY/PARITY DEFAULT	3.7.3 O	RN
RK06 OVERLAPPED SEEK DRIVER TIMEOUTS	3.7.4 M	RN
RK06 DRIVER ERROR LOGGING	3.7.5 M	RN
REWIND COMMAND AT END-OF-TAPE	3.7.6 M	RN

Component	Sequence		
DEVICE DRIVER NOTES			
CR11 CARD READER NOTES	3.8.1 N	RN	
RSTS/E BASIC-PLUS			
BASIC-PLUS PATCHES			
SPECIAL PRINT-USING CHARACTERS	4.1.1 O	RN	
DEFAULT SCALE FACTOR/WARNING MESSAGE	4.1.2 O	RN	
DEFAULT TO EXTEND MODE	4.1.3 O	RN	
COMPILE WITH EXPLICIT EXTENSION	4.1.4 M	RN	
PROGRAMMING HINTS			
HANDLING OF INTEGER AND FLOATING POINT CONSTANTS	4.3.1 N	RN	
SYSTEM UTILITIES PACKAGE			
SHUTUP			
CHANGING 'SHUTUP' SLEEP PARAMETERS	10.21.1 O	RN	
**** WARNING **** DO NOT TRY TO SPEED UP SHUTUP	10.21.2 R	RN	
RSTS/E DATA MANIPULATION PACKAGE			
EDIT			
SPECIAL BASIC-PLUS PROGRAM "PATCH" MODE FOR EDIT	11.3.1 N	RN	
ERROR CONTROL PACKAGE			
PACKAGE NOTES			
ERRLOG.FIL FILE USED IN V06A SHOULD BE DELETED	12.1.1 N	RN	
SPOOLING AND OPERATOR SERVICE PACKAGE			
PACKAGE NOTES			
MANUAL SHUTDOWN OF THE LINE-PRINTER SPOOLING PACKAGE	14.1.1 N	RN	
QUEUE.SYS FILE USED IN V06A SHOULD BE DELETED	14.1.2 N	RN	
SPLRUN			
ELIMINATING EXTRA FORM FEEDS ON SPECIAL FORMS	14.10.1 O	RN	
SYSTEM MANAGEMENT GUIDELINES			
RECOVERY OF DISK DATA			
USING BACKUP ON A TWO DISK SYSTEM	16.1.1 N	RN	
CREATING A SYSTEM DISK RECOVERY MEDIUM	16.1.2 N	RN	
RUN-TIME SYSTEM SUPPORT PACKAGE			
PACKAGE NOTES			
INSTALLING AUXILIARY RUN-TIME SYSTEM SUPPORT	20.1.1 N	RN	

Component	Sequence	
FORTRAN IV/RSTS/E V01C		
FORTRAN IV/PACKAGE NOTES		
FORTRAN IV INSTALLATION PROCEDURE	23.1.1 N	RN
CLARIFICATION: COMPARING ASCII DATE ITEMS	23.1.2 N	RN
UNLOCKING FILES OPENED FOR SHARED ACCESS	23.1.3 N	RN
ENTERING LOWER CASE FROM TERMINAL	23.1.4 N	RN
COMPILER		
CERTAIN "ENCODE/DECODE" STATEMENTS ARE FLAGGED AS SYNTAX ERRORS	23.2.1 M	RN
OTS		
OUTPUTTING REAL NUMBERS	23.3.1 R	RN
RUNTIME SYSTEM		
END-OF-FILE NOT RECOGNIZED FROM KEYBOARD	23.4.1	RN
DMS-500 V02		
NOTES/PROGRAMMING HINTS		
FILE STRUCTURES	24.1.1	RN
UNEXPECTED ERROR ENCOUNTER	24.1.2	RN
DMSFSX.BAS		
EXTENDING EXISTING FILES	24.12.1	RN
IAMGEN		
STRTAD ENTRY IN FDB IS NOT ALWAYS RESET	24.13.1	RN
IAMCRI		
REVISED BLANK RECORD HANDLING	24.14.1	RN
DSORT		
IMPROPERLY HANDLED LOGICAL DEVICE NAMES	24.19.1	RN
MINIMUM RECORD LENGTH FOR "DSORT"	24.19.2	RN
INDIRECT COMMAND FILE CREATION	24.19.3	RN
DSORTD		
DSORT TEST FOR DMS-500 FILE ERRONEOUS	24.20.1	RN
DSORTD MAY RETURN A FATAL ERROR USING MULTIPLE KEY FILES	24.20.2	RN
PREALLOCATION ERROR	24.20.3	RN
DSORTF		
FIELD OVERFLOW ERROR USING /F	24.21.1	RN
DSORTX		
DSORT PRODUCES MISORDERED OUTPUT FILE WITH CORRECT KEYFILE	24.22.1	RN
DSORTS		
MISORDERED OUTPUT FILE ADDENDUM	24.23.1	RN
RSTS/E 2780 PACKAGE		
PACKAGE NOTES		
2780 PACKAGE NOT YET AVAILABLE FOR RSTS/E V06B-02	25.1.1 N	RN

Component	Sequence	
COBOL-11 V02		
COBRG TOO BIG FOR STANDARD COBOL COMPILER	9	RN
CROSS REFERENCE PROBLEM	10	RN
GO TO RESERVE WORD PROBLEM	11	RN
UNSTRING VERB USING NONINTEGER DATA ITEMS	12	RN
COMPILER DOES NOT ACCEPT SOURCE INPUT FROM CARD READER	13	RN
RECORD SIZE APPEARS AS FOUR ASCII BYTES	14	RN
STACK OVERFLOW	17	RN
APPLY MAP8	18	RN
COPY VERB PROBLEM IN "WORKING-STORAGE" SECTION	19	RN
ERRONEOUS "PERFORM" ERROR MESSAGES	20	RN
COMP ITEM SYNCHRONIZATION	21	RN
LIMIT IN PERFORM ITERATION COUNTS	22	RN
TERMINAL FILENAME	23	RN
EFFICIENCY TECHNIQUES	24	RN
RELATIVE I/O	26	RN
COBOL-11 INSTALLATION PROCEDURE	27 N	RN
RPG-II V7.06		
RPG-II RE-RELEASED FOR RSTS/E V06B-02	1 N	RN
SORT-11 V1		
DOCUMENTATION UPDATE	4	RN

RSTS/E V06B-02
RSTS/E INIT.SYS Program
INIT.SYS Program Patches

Seq 1.1.1 M

DC11'S AT PR5 ARE DISABLED - MANDATORY INIT.SYS PROGRAM PATCH

PROBLEM:

When the INIT.SYS program is loaded (booted), it scans all devices on the system for existence. All existing devices are forced to interrupt to determine their vector and interrupt priority. Any device that interrupts at a priority higher than expected is automatically disabled. The DC11 single line, modem control interface normally interrupts at PR5, but the INIT.SYS tables indicate that its maximum priority is PR4. This error causes all DC11's to be disabled.

SOLUTION:

This required patch corrects the problem by changing the INIT.SYS tables to indicate a maximum priority of PR5 for DC11's.

PROCEDURE:

1. This is a required patch to the INIT.SYS Program. Since patching the distribution medium is not recommended, the patch must be installed every time the COPY option is used to copy INIT.SYS from the distribution medium. Any copy of the patched INIT.SYS will propagate the fix.

2. When the COPY option has completed moving all files from the distribution medium, it automatically bootstraps the output disk. The PATCH option should be then used as shown below.

```
Option: PATCH
File to patch? INIT.SYS
Base address? 71472
Offset address? 403
 Base   Offset  Old       New?
071472  000403     000  ? 40
071472  000404     263  ? ^C            (CTRL/C to exit)

Option:
```

RSTS/E V06B-02
RSTS/E INIT.SYS Program
INIT.SYS Program Patches

DZ11'S ARE DISABLED - MANDATORY INIT.SYS PROGRAM PATCH

PROBLEM:

When the INIT.SYS program is loaded (booted), it scans all devices on the system for existence. All existing devices are forced to interrupt to determine their vector and interrupt priority. The routine that forces the DZ11 multiplexor to interrupt does not properly enable interrupts, and, therefore, the DZ11 never interrupts. This error causes all DZ11's to be disabled.

SOLUTION:

This required patch corrects the problem by correctly enabling interrupts on the DZ11 multiplexor.

PROCEDURE:

1. This is a required patch to the INIT.SYS Program. Since patching the distribution medium is not recommended, the patch must be installed every time the COPY option is used to copy INIT.SYS from the distribution medium. Any copy of the patched INIT.SYS will propagate the fix.

2. When the COPY option has completed moving all files from the distribution medium, it automatically bootstraps the output disk. The PATCH option should be then used as shown below.

```
Option: PATCH
File to patch? INIT.SYS
Base address? 25470
Offset address? 4024
 Base    Offset  Old        New?
025470  004024  020000   ? 40040
025470  004026  004736   ? ^C          (CTRL/C to exit)

Option:
```

RSTS/E V06B-02
RSTS/E INIT.SYS Program
INIT.SYS Program Patches

CRASH DUMP/AUTO-RESTART FAILS - MANDATORY INIT.SYS PROGRAM PATCH

PROBLEM:

The crash dump code does not write a valid CRASH.SYS file if the system disk is an RP04/RP05/RP06. In addition, the auto-restart code cannot reboot the INIT.SYS program after a power failure if the system disk is an RK06 or RP04/RP05/RP06.

SOLUTION:

This required patch corrects these problems by altering the crash dump/auto-restart disk driver code.

PROCEDURE:

1. This is a required patch to the INIT.SYS Program. Since patching the distribution medium is not recommended, the patch must be installed every time the COPY option is used to copy INIT.SYS from the distribution medium. Any copy of the patched INIT.SYS will propagate the fix.

2. When the COPY option has completed moving all files from the distribution medium, it automatically bootstraps the output disk. The PATCH option should then be used as shown below.

```
Option: PATCH
File to patch? INIT.SYS
Base address? 60:143750
Offset address? 7256
 Base   Offset  Old       New?
143750  007256  010125  ? 12704
143750  007260  012704  ? 157014
143750  007262  157016  ? 12425
143750  007264  012425  ? ^Z            (CTRL/Z for new offset)
Offset address? ^Z                     (CTRL/Z for new base)
```

```
Base address? 25470
Offset address? 1774
 Base    Offset  Old        New?
025470  001774  013700  ? 4737
025470  001776  157022  ? 25470+5500
025470  002000  012701  ? ^Z            (CTRL/Z for new offset)
Offset address? 5500
 Base    Offset  Old        New?
025470  005500  000000  ? 13700
025470  005502  000000  ? 157022
025470  005504  000000  ? 20027
025470  005506  000000  ? 46504
025470  005510  000000  ? 1003
025470  005512  000000  ? 12737
025470  005514  000000  ? 3
025470  005516  000000  ? 157074
025470  005520  000000  ? 20027
025470  005522  000000  ? 41104
025470  005524  000000  ? 1021
025470  005526  000000  ? 12737
025470  005530  000000  ? 23
025470  005532  000000  ? 157156
025470  005534  000000  ? 12737
025470  005536  000000  ? 22
025470  005540  000000  ? 157166
025470  005542  000000  ? 12737
025470  005544  000000  ? 440
025470  005546  000000  ? 157170
025470  005550  000000  ? 12703
025470  005552  000000  ? 25470+5572
025470  005554  000000  ? 12704
025470  005556  000000  ? 157270
025470  005560  000000  ? 12705
025470  005562  000000  ? 14
025470  005564  000000  ? 12324
025470  005566  000000  ? 77502
025470  005570  000000  ? 207
025470  005572  000000  ? 0
025470  005574  000000  ? 16767
025470  005576  000000  ? 177516
025470  005600  000000  ? 6
025470  005602  000000  ? 10203
025470  005604  000000  ? 5002
025470  005606  000000  ? 6227
025470  005610  000000  ? 0
025470  005612  000000  ? 103731
025470  005614  000000  ? 6303
025470  005616  000000  ? 6102
025470  005620  000000  ? 772
025470  005622  000000  ? ^C            (CTRL/C to exit)

Option:
```

RSTS/E V06B-02
RSTS/E INIT.SYS Program
INIT.SYS Program Patches

REFRESH OPTION FAILS TO DELETE FILES - MANDATORY INIT.SYS PROGRAM PATCH

PROBLEM:

If the FILE suboption of the REFRESH option creates a file or changes a file's size and then attempts to delete another file without returning to the REFRESH SUBOPTION question, the deletion does not succeed.

SOLUTION:

This required patch corrects the problem by changing the initialization code.

PROCEDURE:

1. This is a required patch to the INIT.SYS Program. Since patching the distribution medium is not recommended, the patch must be installed every time the COPY option is used to copy INIT.SYS from the distribution medium. Any copy of the patched INIT.SYS will propagate the fix.

2. When the COPY option has completed moving all files from the distribution medium, it automatically bootstraps the output disk. The PATCH option should be then used as shown below.

```
Option: PATCH
File to patch? INIT.SYS
Base address? 134:117002
Offset address? 1622
 Base   Offset  Old       New?
117002  001622  105767  ? 4737
117002  001624  005704  ? 104226
117002  001626  001012  ? ^Z            (CTRL/Z for new offset)
Offset address? ^Z                      (CTRL/Z for new base)
Base address? 104226
Offset address? 0
 Base   Offset  Old       New?
104226  000000  000000  ? 4737
104226  000002  000000  ? 124136
104226  000004  000000  ? 105737
104226  000006  000000  ? 126534
104226  000010  000000  ? 207
104226  000012  000000  ? ^C            (CTRL/C to exit)

Option:
```

RSTS/E V06B-02
RSTS/E INIT.SYS Program
INIT.SYS Program Patches

UNIBUS WINDOW ALLOCATION - MANDATORY INIT.SYS PROGRAM PATCH

PROBLEM:

The INIT.SYS program allocates UNIBUS WINDOW registers to UNIBUS direct memory access devices on PDP-11/70's with more than 124K words of memory. A mistake in the allocation code causes too many registers to be allocated to each such device. This can cause random system behavior and system crashes.

SOLUTION:

This required patch corrects the problem by fixing the UNIBUS WINDOW allocation code.

PROCEDURE:

1. This is a required patch to the INIT.SYS Program. Since patching the distribution medium is not recommended, the patch must be installed every time the COPY option is used to copy INIT.SYS from the distribution medium. Any copy of the patched INIT.SYS will propagate the fix.

2. When the COPY option has completed moving all files from the distribution medium, it automatically bootstraps the output disk. The PATCH option should then be used as shown below.

```
Option: PATCH
File to patch? INIT.SYS
Base address? 60:143750
Offset address? 6600
 Base   Offset  Old       New?
143750  006600  005200  ? 110025
143750  006602  110025  ? 5200
143750  006604  010403  ? ^C          (CTRL/C to exit)

Option:
```

RSTS/E V06B-02
RSTS/E INIT.SYS Program
INIT.SYS Program Patches

DEVICE CHECKING TIMEOUTS - MANDATORY INIT.SYS PROGRAM PATCH

PROBLEM:

When the INIT.SYS program is loaded (booted), it scans all devices on the system for existence. All existing devices are forced to interrupt to determine their vector and interrupt priority. During this automatic vector determination, INIT.SYS does software timeouts in case some device is malfunctioning or is configured incorrectly. There are several test loops in which a software timeout was mistakenly left out. This can cause INIT.SYS to "hang" forever.

SOLUTION:

This required patch corrects these problems by adding a software timeout to these loops.

PROCEDURE:

1. This is a required patch to the INIT.SYS Program. Since patching the distribution medium is not recommended, the patch must be installed every time the COPY option is used to copy INIT.SYS from the distribution medium. Any copy of the patched INIT.SYS will propagate the fix.

2. When the COPY option has completed moving all files from the distribution medium, it automatically bootstraps the output disk. The PATCH option should then be used as shown below.

RSTS/E V06B-02
RSTS/E INIT.SYS Program
INIT.SYS Program Patches


```
Option: PATCH
File to patch? INIT.SYS
Base address? 25470
Offset address? 2707
 Base   Offset  Old        New?
025470  002707     000  ? 2
025470  002710     000  ? 200
025470  002711     007  ? ^Z            (CTRL/Z for new offset)
Offset address? 3572
 Base   Offset  Old        New?
025470  003572  012701  ? 12762
025470  003574  100000  ? 104
025470  003576  012762  ? 4
025470  003600  000104  ? 12712
025470  003602  000004  ? 1
025470  003604  012712  ? 112762
025470  003606  000001  ? 377
025470  003610  112762  ? 6
025470  003612  000377  ? 4737
025470  003614  000006  ? 25470+5622
025470  003616  004736  ? ^Z            (CTRL/Z for new offset)
Offset address? 4040
 Base   Offset  Old        New?
025470  004040  032712  ? 462
025470  004042  000020  ? ^Z            (CTRL/Z for new offset)
Offset address? 4206
 Base   Offset  Old        New?
025470  004206  032712  ? 12746
025470  004210  000020  ? 100000
025470  004212  001375  ? 137
025470  004214  000207  ? 25470+5640
025470  004216  012700  ? ^Z            (CTRL/Z for new offset)
Offset address? 5622
 Base   Offset  Old        New?
025470  005622  000000  ? 5046
025470  005624  000000  ? 105762
025470  005626  000000  ? 4
025470  005630  000000  ? 100410
025470  005632  000000  ? 5316
025470  005634  000000  ? 1373
025470  005636  000000  ? 405
025470  005640  000000  ? 32712
025470  005642  000000  ? 20
025470  005644  000000  ? 1402
025470  005646  000000  ? 5316
025470  005650  000000  ? 1373
025470  005652  000000  ? 5226
025470  005654  000000  ? 207
025470  005656  000000  ? ^C            (CTRL/C to exit)

Option:
```

RSTS/E V06B-02
RSTS/E INIT.SYS Program
INIT.SYS Program Notes

Seq 1.2.1 N

CLEANING DISKS AFTER SYSTEM CRASHES

In RSTS/E V06B-02, system crash processing has changed somewhat from previous releases:

1. Volatile memory is written to [0,1]CRASH.SYS after every crash if the crash dump facility was enabled at system startup. (Previous versions required console switches set to 177777.)

2. The system performs an automatic restart if the console switches were set to 177777.

If the switches are not set to 177777 at the time of the crash (or the system crashed twice within one minute), the CPU halts at address 54. If, however, auto-restart is enabled, the initialization program INIT.SYS is booted, the system disk cleaned, and the INIT.BAC program entered to perform the commands in the $CRASH.CTL file.

On a multi-disk system, commands in $CRASH.CTL will attempt to mount the remaining disks of the public structure. INIT.BAC automatically executes the CLEAN SYS function call as needed.

There is one disadvantage to using the auto-restart facility after system crashes: because of size constraints, the CLEAN SYS function call is not as extensive as the CLEAN function of the INIT.SYS REFRESH option. Running stand-alone, the latter performs extensive validation of the disk file structure and contains facilities for rebuilding corrupted structures.

It is recommended that systems run with auto-restart enabled to provide fast, unattended crash recovery. Whenever the system is manually shutdown (such as for preventive maintenance), all disks can be checked using the Clean suboption of Refresh. Such checking ensures that no file system inconsistencies exist. This is especially important if there has been a recent system crash and auto-restart.

ADDING BAD BLOCKS TO THE BAD BLOCK FILE

The ERRDIS program prints a list of potential bad blocks. If the system manager decides to add a block to the bad block file, the BADS? suboption of the REFRESH option is used during initialization as described in the System Generation Manual, section 3.7.6.

It is recommended that the following procedure be followed to add a bad block to the bad block file:

1. Copy the data on the bad file to a new file using the /GO switch (along with other necessary switches, such as /CL, etc.):

```
RUN $PIP
#FILE.NEW=FILE.OLD/GO
#FILE.BAD=FILE.OLD/RE
#FILE.OLD=FILE.NEW/RE
```

2. When convenient, shut down the system and use the BADS? suboption to add any referenced bad blocks to the system bad block file. Each new bad block is now allocated to two files ([0,1]BADB.SYS and FILE.BAD in the above example).

3. The CLEAN suboption then rebuilds the allocation tables. CLEAN will indicate that the file has a bad block, allowing the system manager to delete the file (FILE.BAD in the above example). This is described in section 3.7.2 of the System Generation Manual.

When the file is deleted, the non-bad blocks in the file are free for other usage.

Step 1 of the above procedure must be modified slightly if the bad block is in a file directory. In this case, the BACKUP program should be used to copy all accessable files in the account (or disk) to an archive medium. When adding the bad block and cleaning the disk, the CLEAN suboption will allow deleting the account whose directory contains the bad block. After starting time sharing, the system manager may then enter the account using REACT and use BACKUP to restore the files from the archive medium.

RSTS/E V06B-02 Seq 1.2.3 R
RSTS/E INIT.SYS Program
INIT.SYS Program Notes Page 1 of 1

RP04/RP05/RP06 BOOTSTRAP - RESTRICTION

The RP04/RP05/RP06 bootstrap uses the hardware multiply instruction to calculate disk addresses from device cluster numbers. This instruction treats numbers as signed quantities. Since device cluster numbers for the RP04/RP05/RP06 are 16-bit quantities, any calculation using the signed multiply hardware instruction on a device cluster number larger than 32767(decimal) will result in an invalid (i.e. negative) answer. Therefore, the bootstrap code cannot transfer to or from a file containing such a cluster.

Article 1.1.3 patches the RP04/RP05/RP06 bootstrap to fix this problem for the cases in which the bootstrap code is reused: loading the monitor, writing the CRASH.SYS file when the system crashes, and loading stand-alone programs with the LOAD option. However, the bootstrap cannot be patched to work properly when it initially loads the INIT.SYS program from the system disk.

The INIT.SYS program is usually put onto a disk using the COPY option. If that disk can be successfully bootstrapped, you may proceed. If the bootstrap halts at location 157042 and does not recover if you press the CONTinue switch on the console, the INIT.SYS program is probably at a cluster which is too high. You cannot use this disk as a system disk. Reboot the unit from which you used COPY, and use DSKINT to initialize a new RP04/RP05/RP06 pack. Use COPY to load INIT.SYS onto this pack. This ensures that all of the INIT.SYS device cluster numbers are less than 32768.

RSTS/E V06B-02
RSTS/E Executive
Monitor Patches

Seq 3.1.1 0

MAKING RECEIVER DECLARATION NON-PRIVILEGED - OPTIONAL MONITOR PATCH

The RSTS/E monitor normally requires that a job have permanent or temporary privileges in order to declare itself to be a message receiver. There are some production environments in which it is useful for non-privileged jobs to have this capability. The following procedure allows you to patch the RSTS/E monitor to remove the privileged status of the message receiver declaration SYS call.

PROCEDURE:

1. This is an optional patch to the RSTS/E V06B-02 executive. It may be installed in any target monitor SIL.

2. Bootstrap your system disk and use the PATCH option as shown below. If you made the send/receive code resident when you generated the monitor SIL, answer the 'Module name?' question with "FIP". Otherwise, answer with "OVR" as shown.

```
Option: PATCH
File to patch? <lf>      (LINE FEED for installed monitor SIL)
Module name? OVR         (or FIP)
Base address? ..RCVP
Offset address? 0
 Base    Offset  Old       New?
??????  000000  001416  ? 240
??????  000002  010446  ? ^C            (CTRL/C to exit)

Option:
```

RSTS/E V06B-02
RSTS/E Executive
Monitor Patches

CHANGING POWER FAIL DELAY FACTOR - OPTIONAL MONITOR PATCH

When you generated your RSTS/E monitor, you specified a delay factor for power fail recovery. After a power failure, the monitor will wait for this number of seconds before attempting to restart, allowing time for disks to get up to speed.

If the value you specified during SYSGEN is wrong (too long or too short), you can change it by installing the patch given below. There is no need to regenerate.

PROCEDURE:

1. This is an optional patch to the RSTS/E V06B-02 executive. It may be installed in any target monitor SIL.

2. Decide how many seconds the system should delay after power fail. This can be a number from 1 to 300 (decimal). Use this number as the value for n in the patch below. Be sure you include the decimal point after the new value of n so that it is interpreted as a decimal number.

3. Bootstrap your system disk and use the PATCH option as shown below.

```
Option: PATCH
File to patch? <lf>     (LINE FEED for installed monitor SIL)
Module name? RSTS
Base address? ..PWRP
Offset address? 0
 Base   Offset  Old       New?
??????  000000  ??????  ? n.            (from step 2)
??????  000002  012746  ? ^C            (CTRL/C to exit)

Option:
```

RSTS/E V06B-02
RSTS/E Executive
Monitor Patches

Seq 3.1.3 M

MESSAGE SEND/RECEIVE BUFFER HANDLING - MANDATORY MONITOR PATCH

PROBLEM:

If the old message receive monitor call (SYS function code 18.) is used to receive a message which has an extended data portion (i.e. was sent using the extended message send call, SYS function code 22.), the message handling code in the monitor will fail and the system may crash.

In addition, the message send code sometimes fails to return a 16-word buffer to the system.

SOLUTION:

The following required monitor patch fixes these problems.

PROCEDURE:

1. This is a required patch to the RSTS/E V06B-02 executive. It must be installed in all target monitor SILs.
2. Bootstrap your system disk and use the PATCH option as shown below.

RSTS/E V06B-02
RSTS/E Executive
Monitor Patches


```
Option: PATCH
File to patch?  <lf>    (LINE FEED for installed monitor SIL)
Module name? EMT
Base address? CAL
Offset address? 2152
 Base   Offset  Old       New?
120632  002152  012501  ? 4737
120632  002154  012502  ? EMTPAT
120632  002156  001445  ? ^Z            (CTRL/Z for new offset)
Offset address? 2362
 Base   Offset  Old       New?
120632  002362  004537  ? 10004
120632  002364  ??????  ? 137
120632  002366  000763  ? EMTPAT+16
120632  002370  012737  ? ^Z            (CTRL/Z for new offset)
Offset address? ^Z                     (CTRL/Z for new base)
Base address? EMTPAT
Offset address? 0
 Base   Offset  Old       New?
120000  000000  000000  ? 12501
120000  000002  000000  ? 1003
120000  000004  000000  ? 5015
120000  000006  000000  ? 12716
120000  000010  000000  ? CAL+2336
120000  000012  000000  ? 12502
120000  000014  000000  ? 207
120000  000016  000000  ? 4
120000  000020  000000  ? 40
120000  000022  000000  ? 4537
120000  000024  000000  ? REGRES
120000  000026  000000  ? 137
120000  000030  000000  ? CAL+2336
120000  000032  000000  ? ^C            (CTRL/C to exit)

Option:
```

RSTS/E V06B-02
RSTS/E Executive
Monitor Patches

Seq 3.1.4 M

NETWORK MESSAGE SEND FAILURE - MANDATORY MONITOR PATCH

PROBLEM:

The monitor does not distinguish a network message send from a local message send.

SOLUTION:

The following required monitor patch fixes the problem.

PROCEDURE:

1. This is a required patch to the RSTS/E V06B-02 executive. It must be installed in all target monitor SILs.

2. Bootstrap your system disk and use the PATCH option as shown below.

```
Option: PATCH
File to patch?  <lf>    (LINE FEED for installed monitor SIL)
Module name? EMT
Base address? FSS
Offset address? 4056
 Base   Offset  Old       New?
123360  004056  003770  ? 2370
123360  004060  004737  ? ^C            (CTRL/C to exit)

Option:
```

RSTS/E V06B-02 Seq 3.3.1 0
RSTS/E Executive
Terminal Service Patches

RESTRICTING TERMINAL MODES - OPTIONAL TERMINAL SERVICE PATCH

The RSTS/E monitor normally allows any user to OPEN a terminal in binary mode or echo control mode (if configured). There is a danger in the use of these terminal modes, since all characters (including CTRL/C) are passed to the user program. A novice user could easily find himself in a situation in which his terminal does not respond. A malicious user could "lock up" a terminal in a way which required the system manager to kill his job. The following procedure allows you to patch the RSTS/E monitor to specify which terminal OPEN modes require privileges. Privileged users and non-privileged users running privileged programs would still have full access to all available terminal modes. Attempts by non-privileged users to use restricted modes would be rejected with no error message.

PROCEDURE:

1. This is an optional patch to the RSTS/E V06B-02 executive. It may be installed in any target monitor SIL.

2. Decide which terminal modes you wish to make privileged. Consult the RSTS/E Programming Manual for the various modes which are available, and determine the associated MODE values. Compute the sum, n, of the MODE values corresponding to the modes you wish to restrict. If you calculate the sum of the mode values in decimal, include a decimal point after the value of n in the patch below so that it is interpreted as a decimal number.

3. Bootstrap your system disk and use the PATCH option as shown below.

```
Option: PATCH
File to patch? <lf>     (LINE FEED for installed monitor SIL)
Module name? RSTS
Base address? ..KBMP
Offset address? -2
 Base   Offset  Old       New?
??????  177776  042702  ? <lf>          (no change; verify only)
??????  000000  000000  ? n             (from step 2)
??????  000002  ??????  ? ^C            (CTRL/C to exit)

Option:
```

RSTS/E V06B-02
RSTS/E Executive
Terminal Service Patches

IMMEDIATE MODEM HANGUP - OPTIONAL TERMINAL SERVICE PATCH

When the carrier frequency drops on a modem line connecting a terminal to RSTS/E, the monitor normally waits approximately 5 seconds before hanging up the line. This allows recovery from transient line failures or accidental unseating of a phone from an acoustic coupler. In certain areas, especially in the United Kingdom and Europe, computer systems are required to disconnect from such a line within milliseconds. The following patch causes RSTS/E to hang up a modem line as soon as it detects loss of carrier signal on that line.

PROCEDURE:

1. This is an optional patch to the RSTS/E V06B-02 executive. It may be installed in any target monitor SIL that supports modem control.

2. Bootstrap your system disk and use the PATCH option as shown below.

```
Option: PATCH
File to patch? <lf>     (LINE FEED for installed monitor SIL)
Module name? RSTS
Base address? ..UKMD
Offset address? 0
 Base   Offset  Old       New?
??????  000000  000404  ? 240
??????  000002  012704  ? ^C          (CTRL/C to exit)

Option:
```

RSTS/E V06B-02 Seq 3.3.3 0
RSTS/E Executive
Terminal Service Patches Page 1 of 1

CHANGING SPECIAL LOGIN TERMINAL - OPTIONAL TERMINAL SERVICE PATCH

The number of jobs that can log in to a RSTS/E system is limited by the swapping space available, the JOB MAX set at system start-up time, and the login setting (set by the LOGINS, NO LOGINS, and SET LOGINS functions of UTILTY). However, the console terminal (KB0:) is a special terminal, and can log in regardless of the login setting, provided that swapping space and JOB MAX permit.

You can change the number of the special keyboard or disable this feature entirely by installing the patch given below.

PROCEDURE:

1. This is an optional patch to the RSTS/E V06B-02 executive. It may be installed in any target monitor SIL.

2. Decide which terminal should be allowed to log in regardless of the login setting, and use that number as n in the patch below. If you wish to disable this feature, use -1. Be sure you include the decimal point after the new value of n so that it is interpreted as a decimal number.

3. Bootstrap your system disk and use the PATCH option as shown below.

```
Option: PATCH
File to patch? <lf>     (LINE FEED for installed monitor SIL)
Module name? RSTS
Base address? ..CTY.
Offset address? 0
 Base   Offset  Old       New?
??????  000000  000000  ? n.*2 (or -1)  (from step 2)
??????  000002  001404  ? ^C            (CTRL/C to exit)

Option:
```

RSTS/E V06B-02 Seq 3.3.4 0
RSTS/E Executive
Terminal Service Patches

CHANGING A TERMINAL'S INPUT BUFFER QUOTA - OPTIONAL TERMINAL SERVICE PATCH

Inside every terminal's DDB (Device Data Block) is an "input buffer quota" value. The default value is 6. Since there are 30 characters in a buffer, this is equal to 6 times 30 or 180 characters. The terminal service will attempt to buffer that many characters before telling the terminal to stop (by sending it a XOFF). You can increase the input buffer quota value for one or more high speed input terminals by using this optional patch.

NOTE

> There is no guarentee that a terminal will be able to get its full buffer quota. A heavy system load may leave less than the terminal's full buffer quota available.

PROCEDURE:

1. This is an optional patch to the RSTS/E V06B-02 executive. It may be installed in any target monitor SIL.

2. Decide which terminal will have its buffer quota altered, and use that number as n in the patch below. Be sure you include the decimal point after the value of n so that it is interpreted as a decimal number. Also decide what the new buffer quota for that terminal should be, and use that number as m in the patch below. The legal values for m are 6. through about 20. inclusive.

3. Bootstrap your system disk and use the PATCH option as shown below.

```
Option: PATCH
File to patch? <lf>     (LINE FEED for installed monitor SIL)
Module name? RSTS
Base address? KBDDDB
Offset address? DDS.KB*n.+26            (n is the KB number)
 Base   Offset  Old       New?
??????  000026  000006  ? m.  (from step 2)
??????  000030  000000  ? ^C           (CTRL/C to exit)

Option:
```

RSTS/E V06B-02
RSTS/E Executive
Terminal Service Patches

INPUT ESCAPE SEQUENCE HANDLING - OPTIONAL TERMINAL SERVICE PATCH

RSTS/E V06B-02 has been upgraded to allow for multiple character incoming escape sequences. Section 4.4.2 of the RSTS/E Programming Manual (DEC-11-ORPMA-B-D) fully and correctly describes the new escape sequence handling.

One of the new multi-character escape sequences is ESCape followed by "P" followed by one modifier character. On the VT52 terminal, the upper left hand key on the auxiliary keypad (usually blue) generates ESC P. If the terminal user types this key and then any other graphic key, the whole sequence is gathered by the terminal service and delivered to the application program. None of the sequence is echoed. Thus the blue key can be considered a "function" key to be followed by some function code character.

On the other hand, some existing applications using the VT52 may expect this key to generate a standalone escape sequence with no required (or desired) modifier character. You may install the following patch to change the terminal service so that the ESCape, "P" sequence does not expect a modifier character. Note that this patch affects <u>all</u> terminals on the system.

PROCEDURE:

1. This is an optional patch to the RSTS/E V06B-02 executive. It may be installed in any target monitor SIL.

2. Bootstrap your system disk and use the PATCH option as shown below.

```
Option: PATCH
File to patch? <lf>      (LINE FEED for installed monitor SIL)
Module name? RSTS
Base address? TTOCHL
Offset address? 107
 Base    Offset  Old       New?
??????  000107     120  ? 200
??????  000110     001  ? ^C            (CTRL/C to exit)

Option:
```

RSTS/E V06B-02
RSTS/E Executive
Terminal Service Patches

Seq 3.3.6 M

JOB SET-UP FAILURE IF NO SMALL BUFFERS - MANDATORY TERMINAL SERVICE PATCH

PROBLEM:

When the job set-up code in the terminal service cannot create a job because there are no available small buffers, it should print a message (?No logins) on the terminal. In one special set of circumstances, the job set-up code will dispatch to the wrong address and cause a system crash.

SOLUTION:

This required terminal service patch corrects the problem.

PROCEDURE:

1. This is an required patch to the RSTS/E V06B-02 executive. It must be installed in all target monitor SIL's.

2. Answer the initial 'Offset address?' question with 42 (octal) if "echo control" was not included in the terminal service at system generation time. Otherwise answer with 56 (octal).

3. Bootstrap your system disk and use the PATCH option as shown below.

```
Option: PATCH
File to patch? <lf>     (LINE FEED for installed monitor SIL)
Module name? RSTS
Base address? (MODCLK<>0)*12+ENDKEY
Offset address? 42                          (if no "echo control")
        -or-                                (   --see step 2--   )
Offset address? 56                          (if "echo control"   )
 Base   Offset  Old       New?
??????  0000??  004767  ? 4737
??????  0000??  ??????  ? PATCH
??????  0000??  012637  ? ^Z                (CTRL/Z for new offset)
Offset address? ^Z                          (CTRL/Z for new base)
Base address? PATCH
Offset address? 0
 Base   Offset  Old       New?
??????  000000  000000  ? 4737
??????  000002  000000  ? NEWJOB
??????  000004  000000  ? 207
??????  000006  000000  ? ^C                (CTRL/C to exit)

Option:
```

RSTS/E V06B-02 Seq 3.3.7 M
RSTS/E Executive
Terminal Service Patches Page 1 of 1

CONTROL/B IN ECHO CONTROL MODE - MANDATORY TERMINAL SERVICE PATCH

PROBLEM:

If a terminal has incoming escape sequences enabled and is being used in "echo control" mode, the character CONTROL/B is not ignored.

SOLUTION:

This required terminal service patch corrects the problem.

PROCEDURE:

1. This is an required patch to the RSTS/E V06B-02 executive. It must be installed in all target monitor SIL's.

2. Bootstrap your system disk and use the PATCH option as shown below.

```
Option: PATCH
File to patch? <lf>     (LINE FEED for installed monitor SIL)
Module name? RSTS
Base address? ..CTY.
Offset address? 114
 Base   Offset  Old       New?
??????  000114  032761  ? 4737
??????  000116  000040  ? PATCH+6
??????  000120  000010  ? 240
??????  000122  001003  ? ^Z              (CTRL/Z for new offset)
Offset address? ^Z                       (CTRL/Z for new base)
Base address? PATCH
Offset address? 6
 Base   Offset  Old       New?
??????  000006  000000  ? 32761
??????  000010  000000  ? 40
??????  000012  000000  ? 10
??????  000014  000000  ? 1003
??????  000016  000000  ? 132761
??????  000020  000000  ? 10
??????  000022  000000  ? 32
??????  000024  000000  ? 207
??????  000026  000000  ? ^C              (CTRL/C to exit)

Option:
```

RSTS/E V06B-02
RSTS/E Executive
Terminal Service Patches

Seq 3.3.8 M

LEAVING ECHO CONTROL MODE WITH PENDING INPUT - MANDATORY TERMINAL SERVICE PATCH

PROBLEM:

If a job which has its console terminal (or the terminal from which it detached) open in echo control mode closes the channel on which the terminal is open, all characters which have been typed ahead but have not been read by the program are copied to the job's input buffer chain, echoed, and processed. If a CONTROL/C is typed ahead in this situation, it may cause a system crash.

SOLUTION:

This required terminal service patch corrects the problem.

PROCEDURE:

1. This is a required patch to the RSTS/E V06B-02 executive. It must be installed in all target monitor SIL's which were generated with the "echo control" terminal service option.
2. Answer the initial 'Offset address?' question with 54 (octal) if "multi-terminal service" was not included in the terminal service at system generation time. Otherwise answer with 60 (octal).
3. Bootstrap your system disk and use the PATCH option as shown below.

```
Option: PATCH
File to patch? <lf>     (LINE FEED for installed monitor SIL)
Module name? RSTS
Base address? CLS$KB
Offset address? 54                      (if no "multi-terminal service")
       -or-                             (      --see step 2--           )
Offset address? 60                      (if "multi-terminal service"    )
 Base   Offset  Old       New?
??????  0000??  004537  ? <lf>          (no change; verify only)
??????  0000??  ??????  ? PATCH+40
??????  0000??  000040  ? ^Z            (CTRL/Z for new offset)
Offset address? ^Z                      (CTRL/Z for new base)
Base address? PATCH
Offset address? 40
 Base   Offset  Old       New?
??????  000040  000000  ? 116100
??????  000042  000000  ? 3
??????  000044  000000  ? 6300
??????  000046  000000  ? 137
??????  000050  000000  ? FETCH
??????  000052  000000  ? ^C            (CTRL/C to exit)

Option:
```

RSTS/E V06B-02
RSTS/E Executive
Terminal Service Notes

Seq 3.4.1 N

2741 HARDWARE OPTIONS REQUIRED BY RSTS/E

RSTS/E supports 2741-compatible terminals connected through DL11D, DL11E, and DC11 single-line interfaces or DH11 and DZ11 multiplexors. Certain 2741 hardware options are required for proper operation under RSTS/E.

The ability to detect "reverse break" is an option on 2741 terminals supplied by some manufacturers including IBM. A reverse break is a control sequence sent by the computer to a terminal which is currently transmitting (keyboard is unlocked). The purpose of the reverse break is to force the terminal into receive state (keyboard locked). RSTS/E sends a reverse break any time the system has output for the 2741 terminal but internal status tables indicate that the terminal is in transmit state. If the terminal does not recognize the reverse break sequence, lock the keyboard, and switch to receive state, the output will be lost and the terminal can end up in a strange state.

IBM refers to the reverse break detection capability on the IBM 2741 Model 1 Communications Terminal as Feature #4708 - Receive Interrupt. This option must be installed on all IBM 2741 terminals for correct operation under RSTS/E.

The ATTENTION key is also optional on IBM 2741 terminals and is required for operation under RSTS/E. The ATTN key generates a break which is interpreted in several ways by RSTS/E software. The RSTS/E System User's Guide describes the various functions of the ATTN key. IBM refers to the ATTN key and the associated break generation hardware as Feature #7900 - Transmit Interrupt. This option is commonly included on 2741 terminals supplied by IBM and is a standard feature on many 2741-compatible terminals supplied by other manufacturers.

RSTS/E V06B-02
RSTS/E Executive
Terminal Service Notes

2741 TERMINAL INTERFACE WIRING

The DL11 (D's and E's) terminal line interfaces require a special configuration for supporting the 2741. The configuration requires 6 data bits, 1 start bit, 1 stop bit, odd parity, 134.5 baud, and rotary switch position 3 for both input and output (full counter-clockwise is position 1). The following jumpers apply to each condition:

Condition	Jumper	
6 Data bits	NB1	out
	NB2	in
1 Stop bit	2SB	in
	J9	out
	J10	in
	J11	out
Odd parity	NP	in
	EPS	in

The 134.5 baud specification requires a 1.03296M crystal (DEC part #18-05502-6). Refer to the DL11 Installation Procedure for specifications pertinent to the DL11D and DL11E.

RSTS/E V06B-02
RSTS/E Executive
Terminal Service Notes

UNDERSTANDING SYNCHRONIZATION DELAYS

DIGITAL's video terminals (VT50, VT52, etc.) use two special control characters (commonly called XOFF and XON) to request the host computer to suspend and resume data transmission. One use of this synchronization protocol allows the terminal to "freeze" a screen of data giving the user a chance to read it before it has scrolled off the screen. This special mode, called HOLD SCREEN MODE, is enabled and disabled with escape sequences. Once enabled, the terminal detects when a line is about to be scrolled from the screen. The character received from the host computer that would cause the scroll is line feed (LF). At this point the terminal sends an XOFF to the host and awaits the user typing the SCROLL key. During this time the terminal will accept and buffer in a temporary storage area (called a silo) further characters received from the host. Even with assumption that the host computer can cease transmission without any software delays, there are delays in line transmission that cause a determinable number of characters to be received by the terminal after it has sent the XOFF.

HOLD SCREEN MODE is just one use of this synchronization protocol. Another use occurs in the handling of the hard copy option of the VT50 series terminals. The copier is slow (at least compared to the video screen) and the terminal must request host computer transmission suspension to avoid missing data when the copier is running.

The LA180 DECwriter I when connected through a serial line interface (referred to as the LA180S) also uses this synchronization protocol. When the LA180S "gets behind" due to high line transmission speed or carriage restore time, it requests that the host temporarily suspend data transmission. In addition, the printer will send an XOFF when its ONLINE/OFFLINE switch is placed into the OFFLINE position or if it detects an out-of-paper condition.

The RSTS/E terminal service processing of the synchronization protocol is enabled and disabled by the TTYSET commands STALL and NO STALL.

Below is the formula used for calculating the worst case number of characters that would be received after the terminal has sent the XOFF:

$$1 + (3*Y/X) + 2*D*Y$$

> Note: This formula assumes that there is no host computer software delay. In reality there is always some delay at the host computer. This delay might be very small or possibly large depending on system loading and other I/O interrupt activity.

In this formula, X is the terminal's transmission rate in characters per second (CPS), Y is the host computer's transmission rate in CPS, and D is the delay time in seconds for a character to travel from one end of the line to the other.

Considering the formula from left to right:

The 1 is the character storage requirement for the line feed that triggered the XOFF in the first place.

The 3*Y/X is due to the nature of the UART (Universal Asynchronous Receiver/Transmitter) in the terminal. Since the UART is normally double buffered, there can be up to two characters already in the UART's buffers when the terminal decides to send the XOFF. Those two characters plus the XOFF correspond to three character times of terminal to host transmit during which the host may be sending to the terminal. Therefore, it is the ratio of the host and terminal transmission rates that is important for three terminal transmission times.

The 2*D is the total time delay for the line in both directions (the 2) to clear. During this time the host computer may have sent up to 2*D*Y characters to the terminal.

The formula presented here is not completely correct for the VT50 series of video terminals. The VT50 does not double buffer transmission to the host computer in its UART. On the other hand, the formula can be used as a worst case formula to ensure correct operation of the XON/XOFF Synchronization Protocol.

The following tables give the required size of the temporary storage area (silo) given: 1) the transmission speed (baud rate) from the host computer to the terminal, 2) the transmission speed from the terminal to the host computer, and 3) the physical delay of the transmission line. Local terminal connections have essentially a zero delay factor. The Telephone Company specifies a nominal worst case delay of 50ms in a coast-to-coast connection, assuming it is not via satellite.

The horizontal scale of baud rates is the baud rate from the host computer to the terminal.

The vertical scale of baud rates is the baud rate from the terminal to the host computer. Baud rates can be converted to characters per second (CPS) by dividing the baud rate by 10 (1 start bit + 8 data bits + 1 stop bit = 10 bits) with the exception of 110 baud which corresponds to 10 CPS (1 start bit + 8 data bits + 2 stop bits).

Oms Line Delay

	110	150	300	600	1200	2400	4800	9600
110	4	6	10	19	37	73	145	289
150	3	4	7	13	25	49	97	193
300	2	3	4	7	13	25	49	97
600	2	2	3	4	7	13	25	49
1200	1	1	2	3	4	7	13	25
2400	1	1	1	2	3	4	7	13
4800	1	1	1	1	2	3	4	7
9600	1	1	1	1	1	2	3	4

50ms Line Delay

	110	150	300	600	1200	2400	4800	9600
110	5	7	13	25	49	97	193	385
150	4	6	10	19	37	73	145	289
300	3	4	7	13	25	49	97	193
600	3	3	6	10	19	37	73	145
1200	2	3	5	9	16	31	61	121
2400	2	3	4	8	15	28	55	109
4800	2	3	4	7	14	27	52	103
9600	2	3	4	7	13	26	51	100

By referring to the manual that came with your terminal you can determine which baud rate combinations are legal for your terminal.

RSTS/E V06B-02 Seq 3.5.1 0
RSTS/E Executive
File Processor Patches

AUXILIARY LIBRARY ACCOUNTS - OPTIONAL FILE PROCESSOR PATCH

The standard system library for RSTS/E systems is account [1,2]. This account is normally referenced by the $ character in file specifications. The special characters ! % and & can also be used to reference three auxiliary libraries. The default assignments for these characters are [1,3], [1,4], and [1,5] respectively. Although the association of character with account is defined, the three accounts must be created with REACT before the special characters can be used successfully.

In some installations, it may be advantageous to change the account numbers referenced by the three special characters. The account numbers are defined by four words in the RSTS/E monitor. The default account numbers are altered with a simple patch as described below.

PROCEDURE:

1. This is an optional patch to the RSTS/E V06B-02 executive. It may be installed in any target monitor SIL.

2. Decide which account numbers you wish to use for the auxiliary library characters. You may change any or all of the default assignments (although changing the assignment of the $ library character is strongly discouraged). To specify a new project-programmer number of [p,pn], type a new value at the appropriate offset. The new value has the form:

   p.*256.+pn.

   where p is the new project number (followed by a decimal point to ensure that it is interpreted as a decimal number), the multiplication shifts the project to the high byte (again, do not omit the decimal point after the 256), and pn is the new programmer number (followed by a decimal point). p and pn must be between 0 and 254, and p may not be zero.

3. Bootstrap your system disk and use the PATCH option as shown below. In this example, the ! account is patched to [100,250], the % account is left unchanged, and the & account is changed to [1,99]. Of course, you should type in appropriate values for your system.

```
Option: PATCH
File to patch? <lf>     (LINE FEED for installed monitor SIL)
Module name? RSTS
Base address? ..PPN.
Offset address? 0
 Base   Offset  Old       New?
??????  000000  000402  ? <lf>                 (No change to $)
??????  000002  000403  ? 100.*256.+250.       (New value for !)
??????  000004  000404  ? <lf>                 (No change to %)
??????  000006  000405  ? 1.*256.+99.          (New value for &)
??????  000010  ??????  ? ^C                   (CTRL/C to exit)

Option:
```

RSTS/E V06B-02
RSTS/E Executive
File Processor Patches

CHANGING DEFAULT PROTECTION CODE - OPTIONAL FILE PROCESSOR PATCH

When a user creates a file without specifying a protection code for that file, and if the personal default protection code is not set, the RSTS/E monitor creates that file with the system default protection code. This is normally 60., which allows the user read/write access to the file, but restricts all non-privileged users from any access. The legal value range for the default protection code is 0. to 63. inclusive. The following procedure allows you to patch the RSTS/E monitor to change your system's default protection code.

PROCEDURE:

1. This is an optional patch to the RSTS/E V06B-02 executive. It may be installed in any target monitor SIL.

2. Bootstrap your system disk and use the PATCH option as shown below. If you made the disk utility code resident when you generated the monitor SIL, answer the 'Module name?' question with "FIP"; otherwise, answer with "OVR" as shown. Be sure you include the decimal point after the new value so that it is interpreted as a decimal number.

```
Option: PATCH
File to patch? <lf>      (LINE FEED for installed monitor SIL)
Module name? OVR         (or FIP)
Base address? ..DPRT
Offset address? 0
 Base   Offset  Old       New?
??????  000000  000074  ? n.            (the new default)
??????  000002  000001  ? ^C           (CTRL/C to exit)

Option:
```

RSTS/E V06B-02 Seq 3.5.3 0
RSTS/E Executive
File Processor Patches Page 1 of 1

RESTRICTING DISK MODES - OPTIONAL FILE PROCESSOR PATCH

The RSTS/E monitor normally allows any user to OPEN a disk file using any mode except 4096 (decimal) (the read regardless mode), and requires that a user have privileges to open a disk file with MODE 4096. .

You may wish to allow non-privileged users to use mode 4096. or to restrict one or more of the other disk modes (see the RSTS/E Programming Manual for a complete list). The following procedure allows you to patch the RSTS/E monitor to specify which disk file OPEN modes require privileges. Privileged users and non-privileged users running privileged programs would still have full access to all available modes. Attempts by non-privileged users to use restricted modes would be rejected with no error message.

PROCEDURE:

1. This is an optional patch to the RSTS/E V06B-02 executive. It may be installed in any target monitor SIL.

2. Decide which disk file modes you wish to make privileged. Consult the RSTS/E Programming Manual for the various modes which are available, and determine the associated MODE values. Compute the sum, n, of the MODE values corresponding to the modes you wish to restrict. If you wish to restrict special update mode, but leave normal update mode unrestricted, use a value of 4 (not 5). If you wish to restrict both update modes, use 4+1. You may type the addition directly to the new value question of patch. Be sure to include a decimal point after each number so that it is interpreted in decimal.

3. Bootstrap your system disk and use the PATCH option as shown below. If you made the disk utility code resident when you generated the monitor SIL, answer the 'Module name?' question with "FIP"; otherwise, answer with "OVR" as shown.

```
Option: PATCH
File to patch? <lf>      (LINE FEED for installed monitor SIL)
Module name? OVR         (or FIP)
Base address? ..RRNP
Offset address? 0
 Base   Offset  Old       New?
??????  000000  010000  ? n.            (from step 2)
??????  000002  004737  ? ^C            (CTRL/C to exit)

Option:
```

RSTS/E V06B-02 Seq 3.5.4 0
RSTS/E Executive
File Processor Patches

INVERTING DIRECTORY ORDERING - OPTIONAL FILE PROCESSOR PATCH

Normally, when RSTS/E creates a file on disk, the file is added to the User File Directory after all other files for that account on the disk. This causes each disk on the public structure, and every private disk, to be ordered according to creation dates within accounts. This ordering can be changed with the REORDR program.

The following patch allows you to reverse the default ordering of directories by adding files at the beginning of a UFD. Therefore, files which were recently created are nearer to the beginning of the directory. This patch reduces the number of searches required to find a recently created file when it is opened for input. However, if you install this patch, directories will tend to be more scattered on disk and the REORDR program should be run more frequently.

PROCEDURE:

1. This is an optional patch to the RSTS/E V06B-02 executive. It may be installed in any target monitor SIL.

2. Bootstrap your system disk and use the PATCH option as shown below. If you made the disk utility code resident when you generated the monitor SIL, answer the 'Module name?' question with "FIP"; otherwise, answer with "OVR" as shown.

```
Option: PATCH
File to patch? <lf>     (LINE FEED for installed monitor SIL)
Module name? OVR        (or FIP)
Base address? ..LTOP
Offset address? 0
 Base   Offset  Old       New?
??????  000000  011603  ? 5003
??????  000002  004737  ? ^C           (CTRL/C to exit)

Option:
```

RSTS/E V06B-02 Seq 3.5.5 0
RSTS/E Executive
File Processor Patches Page 1 of 1

MAKING FIP SYS CALLS PRIVILEGED - OPTIONAL FILE PROCESSOR PATCH

The following FIP SYS calls are normally non-privileged:

-25. File Attribute Read/Write
-17. Change File's Run-Time System Name
+10. Assign a Device
+13. Zero a Device or Account
+14. Read Accounting Data
+15. Indexed Directory Lookup
+17. Wildcard Directory Lookup

It is possible to make one or more of these SYS calls privileged. Doing so will restrict the use of that SYS call to only privileged users and/or privileged programs. Note that the standard CUSP programs assume that the above SYS calls are non-privileged; unexpected results may occur using the CUSPs if one or more of the above are made privileged.

PROCEDURE:

1. This is an optional patch to the RSTS/E V06B-02 executive. It may be installed in any target monitor SIL.

2. Decide which SYS call code you wish to make privileged. Be sure to include a decimal point after the SYS call code to ensure that it is interpreted in decimal.

3. Bootstrap your system disk and use the PATCH option as shown below.

```
Option: PATCH
File to patch? <lf>     (LINE FEED for installed monitor SIL)
Module name? RSTS
Base address? $UUOTB
Offset address? n.-UU$MIN*2              (from step 2)
 Base   Offset  Old       New?
??????  ??????  ??????  ? Q!1
??????  ??????  ??????  ? ^C             (CTRL/C to exit)

Option:
```

RSTS/E V06B-02 Seq 3.7.1 0
RSTS/E Executive
Device Driver Patches

SPECIAL LINE PRINTER DRUMS - OPTIONAL DEVICE DRIVER PATCH

Most DEC LP11 line printer drums contain a character which corresponds to octal code 040 (the ASCII SPACE character). This character may be a blank, the British Pound Sterling character, or some other optional non-standard character. Normally, this character is irrelevant, since the LP11 is usually wired to print only ASCII codes 041 through 137 (octal) for the 64 character set, or 041 through 176 for the 96 character set. The SPACE character (040), "non-printing" characters (codes 000 to 011, 013, and 016 to 037), and character codes greater than 137 (on printers with the 64 character set) all print as spaces. The space is created by suppressing printer action rather than actually having a hammer strike a character position on the printer drum.

By making a simple wiring change to the LP11, the range of printing characters can be extended to include ASCII code 040. If the printer drum contains a special character in the 040 position, sending an 040 code to the printer will cause that special character to be printed.

In conjunction with this change, you can patch the RSTS/E line printer driver to allow special handling of ASCII 040 on a unit by unit basis. Normally, the driver sends 040 codes (SPACEs) unchanged to the printer. If the wiring change is not installed in the printer, this character will appear as a space regardless of the symbol on the line printer drum. The patch allows you to specify an ASCII code which the driver will translate to 040 before sending it to the printer. The special character will thus be printed. In addition, the driver will translate all 040 codes to 037, which is a non-printing character and appears as a space. (Unlike RSTS/E V06A-02, translation of 040 is performed only if the special character translation is enabled.)

Ideally, you should choose a character which is not commonly used for other purposes but which can be typed and printed on any standard terminal. This allows you to use any standard editor to insert the special character into text destined for the printer. Remember that the conversion in the printer driver overrides the standard symbol (if any) for the selected code. For example, if you select the code 101 (the letter A), the special character is printed in place of any A sent to the driver. It would therefore be impossible to print an A.

All normal printing characters have some meaning in RSTS/E or BASIC-PLUS. For example, the & character starts a comment field in the EXTEND mode of BASIC-PLUS, is an abbreviation for account [1,5], and is a special character in RUNOFF source files. The @ character is seldom used in printed material, but RSTS/E uses it as an assignable account designator.

If none of the printable characters is suitable, one of the ASCII control characters (codes 001 to 037) may be used. For example, CTRL/F (006) can be generated with most terminals and will echo and print as ^F if UP ARROW mode is selected with TTYSET. It has no meaning to any standard DEC terminal.

As system manager, you must choose the character which is appropriate for your installation by studying the ASCII code table and considering your users' needs. The procedure for installing the patch is described below.

PROCEDURE:

1. This is an optional patch to the RSTS/E V06B-02 executive. It may be installed in any target monitor SIL that includes line printer support.

2. Decide which line printer unit should have special character handling. If more than one unit is to be patched, repeat the steps below for each unit you wish to modify.

3. Select the character to represent the special printer symbol. If this character is a printable character (e.g. &), specify the new value in the patch by simply typing

   '&

   and replacing the & with the character you have chosen. If the character is a control character, specify its ASCII code as an octal number (or decimal number followed by a decimal point), or as 'x-100 to represent CTRL/x. For example, CTRL/N may be specified as

   16

   or as

   14.

   or as

   'N-100

   in the patch.

4. Bootstrap your system disk and use the PATCH option as shown below. In this example, the symbol n is used to denote the line printer unit being modified. You should type the actual unit number (between zero and the number of printers -1).

```
Option: PATCH
File to patch? <lf>     (LINE FEED for installed monitor SIL)
Module name? RSTS
Base address? LPDDDB
Offset address? DDS.LP*n+30              (n is the unit number)
 Base   Offset  Old       New?
??????  ??????  000000  ? (new value from step 3)
??????  ??????  000000  ? ^C            (CTRL/C to exit)

Option:
```

5. Install the hardware modification to the printer. The modification is different for the several printer models, as listed below:

   LP01 (2310) Wiring change required to card cage backplane. Remove wire A3A4-36 and connect this wire through a 1K pullup resistor to +5 Volts.

   LP02 (2410) Change to AR16 Data Register Card. Remove card from slot A3A24. Lift Pin 5 on Z15 and tie Pin 5 to ground.

   LP04 (2470) Wiring change required to card cage backplane. Remove wire from A3A24-04 and connect this wire through a 1K pullup resistor to +5 Volts.

   LP05 (2230) Insert jumper W1 on the Logic Control Board 29-21112.

RSTS/E V06B-02
RSTS/E Executive
Device Driver Patches

CHANGING DEFAULT PRINTER FORM LENGTH - OPTIONAL DEVICE DRIVER PATCH

When a line printer is opened, the default hardware form length is set to 66. This is the actual form length for most applications. You can change the default form length by installing the patch given below. The legal form length values are between 7. and 127. inclusive. Note that this patch changes the default form length for all line printers on the system.

PROCEDURE:

1. This is an optional patch to the RSTS/E V06B-02 executive. It may be installed in any target monitor SIL which includes line printer support.

2. Bootstrap your system disk and use the PATCH option as shown below. Be sure you include the decimal point after the new value so that it is interpreted as a decimal number.

```
Option: PATCH
File to patch? <lf>     (LINE FEED for installed monitor SIL)
Module name? RSTS
Base address? ..LP66
Offset address? 0
 Base   Offset  Old       New?
??????  000000  000102  ? n.          (the new default)
??????  000002  112712  ? ^C          (CTRL/C to exit)

Option:
```

3. If you are using the line printer spooling package, you must change the control file (or files) which start the spoolers. Include the /DFLENGTH:n switch in the command line for each spooler to specify the new default form length.

RSTS/E V06B-02 Seq 3.7.3 0
RSTS/E Executive
Device Driver Patches

TU16/TU45 MAGTAPE DENSITY/PARITY DEFAULT - OPTIONAL DEVICE DRIVER PATCH

The default density and parity settings for TU16 or TU45 magtape (device MM) under RSTS/E V06B-02 are 800 BPI and ODD parity. A program can change the density and/or parity value for either file structured or non-file structured operations by using the MAGTAPE function.

You can change your system's default settings for density and/or parity by installing the patch given below. This is useful if you frequently interchange tapes with other systems which use 1600 BPI phase encoded defaults. It is also useful if you want to use the 1600 BPI recording mode as your installation's standard.

NOTE: All RSTS/E V06B-02 distribution magtapes (including optional languages such as COBOL and FORTRAN) are written at 800 BPI with ODD parity. Do not install this patch until after you have completed the system library build procedures, including any optional software.

PROCEDURE:

1. This is an optional patch to the RSTS/E V06B-02 executive. It may be installed in any target monitor SIL which is configured for TU16 or TU45 magtape.

2. Select the desired magtape density and parity. To use 1600 BPI phase encoded mode, specify a new value of 0 in the patch. To return to 800 BPI and ODD parity, specify a new value of 140. Any other value will cause file structured operations to fail.

3. Bootstrap your system disk and use the PATCH option as shown below.

```
Option: PATCH
File to patch? <lf>      (LINE FEED for installed monitor SIL)
Module name? RSTS
Base address? ..MTDP
Offset address? 1
 Base   Offset  Old       New?
??????  000001     140  ? n              (from step 2)
??????  000002     ???  ? ^C             (CTRL/C to exit)

Option:
```

RSTS/E V06B-02 Seq 3.7.4 M
RSTS/E Executive
Device Driver Patches Page 1 of 1

RK06 OVERLAPPED SEEK DRIVER TIMEOUTS - MANDATORY DEVICE DRIVER PATCH

PROBLEM:

The RK611/RK06 overlapped seek disk device driver fails to reenable controler interrupts in certain cases. This can cause the logging of timeout errors on one or more drives.

SOLUTION:

The following required device driver patch fixes the problem.

PROCEDURE:

1. This is an required patch to the RSTS/E V06B-02 executive. It must be installed in any target monitor SIL which is configured for RK06 disks using the overlapped seek driver.

2. Bootstrap your system disk and use the PATCH option as shown below.

```
Option: PATCH
File to patch? <lf>     (LINE FEED for installed monitor SIL)
Module name? RSTS
Base address? DMSEEK
Offset address? 234
 Base   Offset  Old       New?
??????  000234  001414  ? 240
??????  000236  012764  ? ^Z           (CTRL/Z for new offset)
Offset address? 366
 Base   Offset  Old       New?
??????  000366  000005  ? 105
??????  000370  004737  ? ^C           (CTRL/C to exit)

Option:
```

RSTS/E V06B-02
RSTS/E Executive
Device Driver Patches

RK06 DRIVER ERROR LOGGING - MANDATORY DEVICE DRIVER PATCH

PROBLEM:

The RK611/RK06 disk device driver (both non-overlapped seek and overlapped seek) fails to log all possible device error information in certain cases.

SOLUTION:

The following required device driver patch fixes the problem.

PROCEDURE:

1. This is an required patch to the RSTS/E V06B-02 executive. It must be installed in any target monitor SIL which is configured for RK06 disks.

2. Bootstrap your system disk and use the PATCH option as shown below.

```
Option: PATCH
File to patch? <lf>      (LINE FEED for installed monitor SIL)
Module name? RSTS
Base address? DMDSK
Offset address? 44
 Base    Offset  Old       New?
??????  000044  016703  ? 4737
??????  000046  ??????  ? PATCH+26
??????  000050  005002  ? ^Z            (CTRL/Z for new offset)
Offset address? ^Z                     (CTRL/Z for new base)
Base address? PATCH
Offset address? 26
 Base    Offset  Old       New?
??????  000026  000000  ? 13703
??????  000030  000000  ? CSR.DM
??????  000032  000000  ? 12713
??????  000034  000000  ? 100000
??????  000036  000000  ? 207
??????  000040  000000  ? ^C            (CTRL/C to exit)

Option:
```

RSTS/E V06B-02
RSTS/E Executive
Device Driver Patches

REWIND COMMAND AT END-OF-TAPE - MANDATORY DEVICE DRIVER PATCH

PROBLEM:

The TU16/TU45 magtape device driver returns the end-of-tape error (?No room for user on device; ERR code 4) if a rewind command (magtape special function code 3) is issued when the tape is positioned at the end-of-tape reflector. The rewind command does rewind the tape, but the error should not be given.

SOLUTION:

The following required device driver patch fixes the problem.

PROCEDURE:

1. This is an required patch to the RSTS/E V06B-02 executive. It must be installed in any target monitor SIL which is configured for TU16/TU45 magtape.

2. Bootstrap your system disk and use the PATCH option as shown below.

```
Option: PATCH
File to patch? <lf>     (LINE FEED for installed monitor SIL)
Module name? RSTS
Base address? MMDVR
Offset address? 710
 Base   Offset  Old       New?
??????  000710  010564  ? 4737
??????  000712  000020  ? PATCH+52
??????  000714  012702  ? ^Z              (CTRL/Z for new offset)
Offset address? ^Z                       (CTRL/Z for new base)
Base address? PATCH
Offset address? 52
 Base   Offset  Old       New?
??????  000052  000000  ? 126427
??????  000054  000000  ? 16
??????  000056  000000  ? 4
??????  000060  000000  ? 1002
??????  000062  000000  ? 42705
??????  000064  000000  ? 2000
??????  000066  000000  ? 10564
??????  000070  000000  ? 20
??????  000072  000000  ? 207
??????  000074  000000  ? ^C              (CTRL/C to exit)

Option:
```

RSTS/E V06B-02
RSTS/E Executive
Device Driver Notes

CR11 CARD READER NOTES

The CR11 card reader normally performs a dark check at column 81 of each card. Cards which have been verified using the IBM 129 punch/verifier or similiar devices will have a verification punch in column 81. This will result in a READ CHECK condition.

There is no way to fix this problem in software since the READ CHECK must be cleared manually at the card reader. There is, however, a hardware fix.

The presence or absence of a wire jumper between pad A and pad B on the "CLOCK" card inside the CR11 reader selects either column 0 or column 81 for the dark check.

Refer to the Documation M200 Card Reader Manual for the clock card schematic and the location of this printed circuit card in the reader.

Jumper A-B

In	Selects dark check at Column 0
Out	Selects dark check at Column 81

The jumper should be present (in) if cards are read which have verification punches in column 81.

RSTS/E V06B-02
RSTS/E BASIC-PLUS
BASIC-PLUS Patches

SPECIAL PRINT-USING CHARACTERS - OPTIONAL BASIC-PLUS PATCH

The Print-Using option of BASIC-PLUS provides for floating dollar sign, comma insertion every three digits to the left of the decimal point, and, of course, printing of the decimal point. The $-sign is used for the currency symbol in the United States and Canada, but various other symbols are used for other monetary systems. In England it would be useful to float the Pound Sterling symbol (or whatever ASCII character is used to represent that symbol) instead of the $-sign. In France a floating Franc symbol (or F) would be generally more useful than the floating $-sign. The role of the comma and period are also reversed in France (e.g. 9999999.02 should be printed as F9.999.999,02).

To accommodate these differences in monetary systems, Print-Using is coded to allow substitution of any character for the floating currency symbol, the decimal point character, and the "every three digits" character. These three characters are defined by three words in the BASIC-PLUS Run-Time System. System defaults and several suggestions are shown in the table below:

Location	Usage	Default	England	France
..PUCH+0	Floating currency symbol	$ (044)	& (046)	F (106)
..PUCH+2	Decimal point character	. (056)	. (056)	, (054)
..PUCH+4	Every three digits character	, (054)	, (054)	. (056)

Substitution of different characters, such as those listed under England and France above, is done with a simple patch. Any change will affect both the Print-Using format string and the output produced. For example, if the suggestions under "France" were installed, the following Print-Using statement would result in the output shown:

```
PRINT USING "FF#.######,##", 3673298.02, 4545.20

F3.673.298,02
    F4.545,20
```

Since any change in this area does render the Print-Using documentation slightly incorrect, an appropriate note should be published for users of the system.

PROCEDURE:

1. This is an optional patch to the BASIC-PLUS Run-Time System. It may be installed in any BASIC-PLUS Run-Time System configured with Print-Using.

2. Determine the three special characters you wish to use for the floating currency symbol, the decimal point character, and the "every three digits" character. The chosen characters can be entered as either:

   'x where x is the chosen printable character
   n where n is the octal value of the chosen character
   n. where n is the decimal value of the chosen character

3. Bootstrap your system disk and use the PATCH option as shown below. Answer the 'File to patch?' question with the name of the BASIC-PLUS Run-Time System to be patched.

```
Option: PATCH
File to patch? BASIC.RTS        (BASIC-PLUS Run-Time System name)
Base address? ..PUCH
Offset address? 0
 Base   Offset  Old       New?
??????  000000  000044  ? 'F          (currency sign from step 2)
??????  000002  000056  ? ',         (decimal point from step 2)
??????  000004  000054  ? '.         ("every three" from step 2)
??????  000006  ??????  ? ^C         (CTRL/C to exit)

Option:
```

DEFAULT SCALE FACTOR/WARNING MESSAGE - OPTIONAL BASIC-PLUS PATCH

The BASIC-PLUS default SCALE factor may be modified by utilizing the procedure in this article (first optional patch). In addition, the warning message (%SCALE Factor Interlock) associated with SCALE may be deleted (second optional patch).

PROCEDURE:

1. This is an optional patch to the BASIC-PLUS Run-Time System. It may be installed in any BASIC-PLUS Run-Time System configured with the 4-word, scaled math packages.

2. Determine the new BASIC-PLUS default SCALE factor. This default SCALE factor must be between 0 and 6 inclusive.

3. Bootstrap your system disk and use the PATCH option as shown below. Answer the 'File to patch?' question with the name of the BASIC-PLUS Run-Time System to be patched.

```
Option: PATCH
File to patch? BASIC.RTS        (BASIC-PLUS Run-Time System name)
Base address? ..SCA.
Offset address? 0
 Base   Offset  Old       New?
??????  000000  000000  ? n             (from step 2)
??????  000002  ??????  ? ^C            (CTRL/C to exit)

Option: PATCH
File to patch? BASIC.RTS        (BASIC-PLUS Run-Time System name)
Base address? ..SCE.
Offset address? 0
 Base   Offset  Old       New?
??????  000000  104577  ? 240           (for no warning message)
??????  000002  000207  ? ^C            (CTRL/C to exit)

Option:
```

RSTS/E V06B-02 Seq 4.1.3 0
RSTS/E BASIC-PLUS
BASIC-PLUS Patches Page 1 of 1

DEFAULT TO EXTEND MODE - OPTIONAL BASIC-PLUS PATCH

Normally BASIC-PLUS defaults to the NO EXTEND mode. To change this default to EXTEND mode apply the patch in this article.

PROCEDURE:

1. This is an optional patch to the BASIC-PLUS Run-Time System. It may be installed in any BASIC-PLUS Run-Time System.

2. Bootstrap your system disk and use the PATCH option as shown below. Answer the 'File to patch?' question with the name of the BASIC-PLUS Run-Time System to be patched.

```
Option: PATCH
File to patch? BASIC.RTS        (BASIC-PLUS Run-Time System name)
Base address? ..XTN.
Offset address? 0
 Base   Offset  Old       New?
??????  000000  000000  ? 20000
??????  000002  ??????  ? ^C            (CTRL/C to exit)

Option:
```

The BASIC-PLUS programs supplied by DIGITAL generally assume a default mode of NO EXTEND. Therefore, when building a library or modifying a CUSP, the NO EXTEND mode should be in effect, unless otherwise noted.

RSTS/E V06B-02
RSTS/E BASIC-PLUS
BASIC-PLUS Patches

COMPILE WITH EXPLICIT EXTENSION - MANDATORY BASIC-PLUS PATCH

PROBLEM:

The BASIC-PLUS COMPILE command defaults the compiled file's extension to .BAC. If an explicit extension is typed to the COMPILE command this default should be overridden but it is not.

SOLUTION:

The following required BASIC-PLUS patch fixes the problem.

PROCEDURE:

1. This is a required patch to the BASIC-PLUS Run-Time System. It must be installed in all BASIC-PLUS Run-Time Systems.

2. Bootstrap your system disk and use the PATCH option as shown below. Answer the 'File to patch?' question with the name of the BASIC-PLUS Run-Time System to be patched.

```
Option: PATCH
File to patch? BASIC.RTS        (BASIC-PLUS Run-Time System name)
Base address? ED
Offset address? 4060
 Base   Offset  Old       New?
??????  004060  012764  ? 4737
??????  004062  000036  ? PA
??????  004064  000004  ? 1003
??????  004066  013764  ? ^Z            (CTRL/Z for new offset)
Offset address? ^Z                     (CTRL/Z for new base)
Base address? PA
Offset address? 0
 Base   Offset  Old       New?
??????  000000  000000  ? 12764
??????  000002  000000  ? 36
??????  000004  000000  ? 4
??????  000006  000000  ? 5764
??????  000010  000000  ? 14
??????  000012  000000  ? 207
??????  000014  ??????  ? ^C            (CTRL/C to exit)

Option:
```

RSTS/E V06B-02
RSTS/E BASIC-PLUS
Programming Hints

HANDLING OF INTEGER AND FLOATING POINT CONSTANTS

The V06B-02 BASIC-PLUS handles integer and floating point constants differently than all previous versions.

INTEGER CONSTANTS

Former versions of BASIC-PLUS generated 5 bytes of code and data storage for every reference to an integer constant. These were: 1 byte of pseudo-opcode, 2 bytes of address, and 2 bytes of data (the constant itself). The V06B-02 version of BASIC-PLUS generates 3 bytes of code and data for all integer constant references: 1 byte of pseudo-opcode and 2 bytes of data (the constant itself). It also treats the integers zero (0%) and one (1%) as special cases. These commonly used integers generate only 1 byte of code, a special pseudo-opcode. One programming technique in earlier versions of BASIC-PLUS was to assign some variable permanently to a constant value and use that variable instead of the literal constant throughout the program (variable references generated only 3 bytes of code). With the enhancement of BASIC-PLUS, this technique is no longer necessary and changing a program to use literal integer constants may well save further space!

FLOATING POINT CONSTANTS

Each reference to a floating point constant normally generates 7 (or 11 for 4-word math) bytes of code and data storage: 1 byte of pseudo-opcode, 2 bytes of address, and 4 (or 8) bytes of data (the constant itself). The V06B-02 version of BASIC-PLUS checks the value of every floating point constant before generating the above sequence of code and data. If the constant has no non-zero bits in its least significant word (or 3 words for 4-word math), BASIC-PLUS generates an alternate 3 byte sequence: 1 byte of special pseudo-opcode and 2 bytes of data (the leading or most significant word of the constant). Most commonly used floating point numbers fall into this special class. Furthermore, BASIC-PLUS will recognize the floating point constants zero (0.) and one (1.) and generate only 1 byte of special pseudo-opcode.

RSTS/E V06B-02
System Utilities Package
SHUTUP

Seq 10.21.1 0

CHANGING 'SHUTUP' SLEEP PARAMETERS - OPTIONAL CUSP PATCH

PROBLEM:

In the "Initial Job Killing Phase", SHUTUP makes two passes through the system job table looking for active jobs which can be logged off the system or KILLed. For those jobs which are attached to a keyboard, the following text string is FORCEd to the terminal's input buffer to simulate the log off sequence:

```
CTRL/C  CTRL/C  "BYE/Y" <cr>
```

Since this method requires the running of the system LOGOUT program, the actual logging off process can consume a fair amount of elapsed time after the text string FORCE is made.

The SHUTUP program is initially set up with the following parameters which control a program pause (SLEEP) at the end of the job table search pass:

```
SLEEP.BASE%      = 10%
SLEEP.INCREMENT% =  2%
```

The length of the SLEEP period in seconds is calculated as:

```
SLEEP time = SLEEP.BASE% + (SLEEP.INCREMENT% * <number of terminals
                                                "killed">)
```

where the number of terminals "killed" is the total number of jobs sent the FORCE text string during the pass through the job table. For example, for 3 terminals, a period of 16 seconds results; for 10 terminals, a period of 30 seconds results.

Some installations may desire to change these parameters to alter the SLEEP period during this phase of SHUTUP operation. A case in point is when a system has at least one terminal running on the system at 110 baud. At this transmission speed, it requires 25-30 seconds just to type out the LOGOUT message which appears after typing BYE/Y. Changing the value of SLEEP.BASE% to 30% should eliminate this problem.

SOLUTION:

Below is presented a procedure for changing the value(s) SLEEP.BASE% and/or SLEEP.INCREMENT%.

Assume the following parameter values for the edit session which follows:

```
xx = new SLEEP.BASE%      value
yy = new SLEEP.INCREMENT% value
```

PROCEDURE:

1. For purposes of the discussion, we will assume that the required program to be patched is located on the public disk structure. In the patching procedure, we will refer to the program by its most simple name format: <program name>.BAS. The person performing the patching operation should, if necessary, replace all program references with suitable text according to the requirements of his installation.

   Also, for purposes of presenting the patching procedure, we assume that the patching operation will be performed in a privileged account.

2. To apply the indicated corrections, perform the following RSTS/E system commands.

```
RUN $EDIT<cr>
EDIT   V06B-03 RSTS V06B-02 Timesharing

#SHUTUP.BAS=SHUTUP.BAS/PATCH<cr>

*H/1031<tab>/V<cr>
1031        SLEEP.BASE% = 10%
*G/10/-2C/xx/V<cr>
1031        SLEEP.BASE% = xx%
*AV<cr>
        \ SLEEP.INCREMENT% = 2%
*G/2/-C/yy/V<cr>
        \ SLEEP.INCREMENT% = yy%
*EX<cr>

#^Z

Ready
```

3. To re-compile the program and re-enter it into the system library type the following RSTS/E commands.

```
OLD SHUTUP<cr>

Ready

COMPILE $

Ready
```

4. The person making the changes to the program should now take whatever steps are necessary, according to installation guidelines, to save the new version of the program.

5. The source (.BAS) version of the program should now be removed from the public structure by using either the UNSAVE or the KILL commands of RSTS/E.

RSTS/E V06B-02
System Utilities Package
SHUTUP

Seq 10.21.2 R

**** WARNING **** DO NOT TRY TO SPEED UP SHUTUP - RESTRICTION

In the V06A-02 release, the speed with which the SHUTUP program brought the RSTS/E system down was influenced (not always reliably) by typing carriage returns at the system console terminal while SHUTUP was going through its various phases. Although this was an unsupported use of SHUTUP, it has become a wide-spread "habit".

The V06B-02 version of SHUTUP attempts to bring down the RSTS/E system in the fastest manner possible. It has been carefully designed for speed, and any attempt to speed it up by typing <cr> on the system console terminal will result in unpredictable operation of SHUTUP, including failure of the SHUTUP program and possible loss of user data on disks that are not shutdown properly.

%%%%% WARNING %%%%%

DO NOT TYPE UNSOLICITED <cr> CHARACTERS TO 'SHUTUP'

TO TRY TO SPEED IT UP.

%%%%% WARNING %%%%%

RSTS/E V06B-02
RSTS/E Data Manipulation Package
EDIT

SPECIAL BASIC-PLUS PROGRAM "PATCH" MODE FOR EDIT

Because some of the BASIC-PLUS programs released in RSTS/E V06B-02 require large amounts of memory when OLDed or RUN, it has become difficult, if not impossible in some cases, to issue Patch Notices for these CUSPs which utilize the standard BASIC-PLUS editor functions. As a result, the standard procedures for patching CUSPs will utilize the EDIT text editing program. To facilitate these procedures, a special option has been included in EDIT and is invoked by appending the switch /PATCH to the EDIT program input command line.

To illustrate, several examples follow:

```
EDIT FOO.BAS/PATCH
EDIT FOO.FIL=ANYFOO.TXT/PATCH
RUN $EDIT
#A.B=C.D/PATCH
```

Since BASIC-PLUS uses the <lf> character to signify that the next text line is a continuation of the current text line (in the NO EXTEND mode and the EXTEND mode if not using the &<cr> continuation convention), there is a conflict in using the EDIT program on CUSP source files because of EDIT's use of <lf> to end insert and change text argument strings. The new option, when invoked, causes EDIT to use the <esc> character in place of the <lf> character to terminate these strings, thus allowing the <lf> character to be used like any other insertable character.

For maintenance of all of the RSTS/E V06B-02 CUSP programs, the published Patch Notices will employ a procedure using the EDIT program and the /PATCH option.

RSTS/E V06B-02
Error Control Package
Package Notes

ERRLOG.FIL FILE USED IN V06A SHOULD BE DELETED

Because of structural changes to the ERRLOG.FIL file required due to the increased functionality of the error logging programs, the V06A version of ERRLOG.FIL should be deleted prior to the initial running of the V06B ERRINT/ERRCPY. This will avoid any conflict that may be caused by structural inconsistencies between the two versions. ERRCPY will create and initialize a new ERRLOG.FIL file when it determines that the file is not present.

MANUAL SHUTDOWN OF THE LINE-PRINTER SPOOLING PACKAGE

On occasion it may become necessary to shutdown the SPOOL program without shutting down the entire system. There are several ways to perform the shutdown.

At any time, an operator may issue the END command to the spooler. This command conditions the spooler to shut itself down (kill itself) the next time it is ready to get a new job. Thus, if the spooler is printing a job, it will complete the job's output before shutting down. If the spooler is waiting for a job, it will shut down immediately. If the operator doesn't want the job currently printing to complete, the ABORT command may be used to terminate the job.

A second method of shutting down the spooler is through an OFFLINE command. This command causes immediate termination of any job being printed with an ABORT message on the listing. It also causes the END processing, and the spooler will kill itself after the current job.

If the operator wishes to terminate a printing job for shutdown or any other reason, the line printer must be online. Under no circumstances will the spooler consider a job completed until all the data which the spooler has tried to print (including termination messages) has been actually printed. If a job is ABORTed by the operator, the spooler will clear out, at the first opportunity, whatever data has already been buffered; the line printer must be online to allow this. If it is desired that no further output be done, it is necessary to use the UTILTY program to KILL the spooler job. In this case, no cleanup can be performed. This process may be necessary in case of certain hardware failures (for example, if the line printer interface will not raise the READY flag). It should only be used in a crisis. The job which was being printed should be killed from the queue file by a QUE/K command from the operator.

For example, to manually shut down the LP0SPL spooler in logical end mode, the following command is used:

```
PLE/INT LP0SPL:END
```

At the completion of the current job, the spooler will kill itself. To manually shut down the BA0SPL job in immediate mode, the command:

```
PLE/INT BA0SPL:OFFLINE
```

is used. BA0SPL will ABORT the current job, clear it from QUEUE.SYS, and kill itself.

RSTS/E V06B-02 Seq 14.1.2 N
Spooling and Operator Services Package
Package Notes Page 1 of 1

QUEUE.SYS FILE USED IN V06A SHOULD BE DELETED

Because of structural changes to the QUEUE.SYS file required due to the increased functionality of the QUE and QUEMAN programs, the V06A version of QUEUE.SYS should be deleted prior to the initial running of the V06B version of QUEMAN. This will avoid any conflict that may be caused by structural inconsistencies between the two versions. QUEMAN will create and initialize a new QUEUE.SYS file when it determines that the file is not present.

RSTS/E V06B-02 Seq 14.10.1 0
Spooling and Operator Services Package
SPLRUN

ELIMINATING EXTRA FORM FEEDS ON SPECIAL FORMS - OPTIONAL CUSP PATCH

PROBLEM:

At the beginning of each job, the spooler performs two top of form operations. The purpose of this action is to ensure that the new job really starts at the top of a new form, with at least one blank page between it and the last job. When using special paper forms, however, it is usually necessary to do a forms alignment. In this case, it may be annoying to have the extra top of form executed because it wastes a page of the form.

SOLUTION:

The following patch inhibits the extra top of form operation at the start of a new job. Note that it inhibits the action not only after a forms alignment process, but also at the beginning of any job. If this patch is used, it is essential that the paper be very carefully aligned in the printer. If the paper is misaligned, one job will begin on the same physical page as the end of the previous job.

PROCEDURE:

1. For purposes of the discussion, we will assume that the required program to be patched is located on the public disk structure. In the patching procedure, we will refer to the program by its most simple name format: <program name>.BAS. The person performing the patching operation should, if necessary, replace all program references with suitable text according to the requirements of the installation. Also, for purposes of presenting the patching procedure, we assume that the patching operation will be performed in a privileged account.

2. To apply the indicated corrections, perform the following RSTS/E system commands.

```
RUN $EDIT<cr>
EDIT    V06B-03 RSTS V06B-02 Timesharing

#SPLRUN.BAS=SPLRUN.BAS/PATCH<cr>

*H/3200/V<cr>
3200    GOSUB 14500 IF PRTLNE%<>1%
*AV<cr>
        \ NXTLNE%=2% \ RECLEN%=0% \ GOSUB 14800
*2KV<cr>
        \ GOSUB 21000
*EX<cr>

#^Z

Ready
```

3. To re-compile the program and re-enter it into the system library, type the following RSTS/E commands.

```
OLD SPLRUN<cr>

Ready

COMPILE $

Ready
```

4. The person making the changes to the program should now take whatever steps are necessary, according to installation guidelines, to save the new version of the program.

5. The source (.BAS) version of the program should now be removed from the public structure by using either the UNSAVE or the KILL commands of RSTS/E.

RSTS/E V06B-02
System Management Guidelines
Using BACKUP on a Two Disk System

USING BACKUP ON A TWO DISK SYSTEM

This article describes the procedure for using the RSTS/E V06B BACKUP package to back up the disks on a system which has only two disk drives and no magtape drive. On such a system, you can back up the system disk normally by dismounting the nonsystem disk. The only requirement is that the BACKUP package exist entirely on the system disk. However, it is impossible for you to take a normal backup of the nonsystem disk. Such a procedure would require that the system disk, the disk being backed up, and the backup volume all be mounted simultaneously, and hence would require three or more disk drives.

The solution presented in this article is to place a minimal system on the other disk, allowing it to run as the system disk during the backup. You must follow this procedure for every RSTS/E file structured disk pack, public or private, which you ever mount along with the system disk and which you want to be able to back up. This procedure has one major drawback: the minimal system requires a large number of blocks on the nonsystem disk (approximately 1400). If you have RK05 disks, you should not use this procedure. Instead, use ROLLIN to preserve RK05 disks. If you have RK06 disks, carefully consider the consequences of using almost 6% of your nonsystem disk capacity for backup. You may wish to make the dangerous choice of not backing up your nonsystem disk(s) and thereby risking loss of data from hardware or software failure, or you could consider expanding your hardware to include another disk drive or a magtape unit. For disks larger than RK06's, the overhead of having a small system on each nonsystem disk is usually small enough to be tolerable, although you must decide based on your system's needs.

If you decide to build a small system on each nonsystem disk, you may use your standard monitor SIL, the SYSGEN.SIL provided by Digital, or another SIL generated especially for the purpose. The last choice allows you to build a small, special-purpose monitor for use only during backup and restore, thereby minimizing the disk space required. If you generate a special monitor, it should support at least 2 jobs, your two disks, one terminal (KB0:), have nothing resident, no options, and a minimal number of small buffers. Remember to install any applicable patches to the SIL.

You should place the minimal system for backup on the nonsystem disk immediately after initializing the disk with DSKINT. This placement ensures that contiguous space is available for the required files. After generating the monitor, Run-Time System, and system library (including the BACKUP package), perform the procedures outlined below. In the examples, the notation 'sys:' denotes the disk drive on which your normal system disk is mounted, and 'xxn:' denotes the other disk drive on the system, which is used for various purposes in the procedure.

1. Ensure that the monitor SIL you want to use for your small backup system is on the system disk. Generate it or copy it if necessary.

2. Ensure that the RT11 Run-Time System is on your system disk in account [0,1], and that the programs PIP.SAV and HOOK.SAV are on your system disk in account [1,2]. If not, mount your distribution system disk (with /RONLY) on xxn: and use the following commands to copy RT11 and the required system library programs (computer responses are underlined):

```
RUN $PIP
PIP    V06B-03 RSTS V06B-02 Your system
#SY0:[0,1]RT11.RTS/CL:32/BL=xxn:[0,1]RT11.RTS
#SY0:$HOOK.SAV/BL=xxn:$HOOK.SAV
#SY0:$PIP.SAV/BL=xxn:$PIP.SAV
#SY0:$HOOK.SAV<124>/RE,$PIP.SAV<104>
#^Z

Ready

RUN $UTILTY
UTILTY  V06B-03 RSTS V06B-02 Your system
#NAME RT11=$HOOK.SAV
#NAME RT11=$PIP.SAV
#^Z

Ready
```

3. If you have not yet initialized the nonsystem disk you plan to use, shut down the system, mount the disk on xxn:, and use DSKINT to initialize the disk. Start timesharing again.

4. Use UTILTY to ADD RT11 as a Run-Time System if you have not already done so.

5. Issue the commands given below:

```
RUN $UTILTY
UTILTY  V06B-03 RSTS V06B-02 Your system
#MOUNT xxn:packid/PRIVATE
#UNLOCK xxn:
#^Z

Ready

RUN $PIP.SAV
*xxn:[0,1]*.*/MO:16=[0,1]INIT.SYS,[0,1]ssssss.SIL
*xxn:[0,1]*.*/MO:16=[0,1]bbbbbb.RTS,[0,1]ERR.ERR
     where ssssss is the SIL you wish to use on your backup disk, and
     bbbbbb is the BASIC-PLUS Run-Time System under which the BACKUP
     package was compiled
*xxn:$*.*/W=SY0:$BAC???.BAC,$BACKUP.PRM,$SHUTUP.BAC
     to copy the BACKUP package to the nonsystem disk
*^Z
```

```
Ready

OPEN 'xxn:[0,1]SWAP.SYS/MO:16/SI:128' FOR OUTPUT AS FILE 1%

Ready

CLOSE 1%

Ready

RUN $HOOK.SAV
*xxn:[0,1]INIT.SYS
Hook complete
*^Z

Ready
```

6. Shut down your system, and use the BOOT option of the initialization code to bootstrap the disk xxn:. Use INSTALL to install the ssssss SIL, which you copied in step 5. Use DEFAULT to set defaults for the backup system. Specify 16K as the SWAP MAX, and bbbbbb as the default Run-Time System.

Your nonsystem disk is now ready to use. You may dismount it and mount it as a public disk if you desire. When you wish to back up that disk, you must shut down your system and bootstrap the nonsystem disk. Use START to start timesharing from the small system on the nonsystem disk, but do not allow INIT.BAC to run. Physically remove your normal system disk from sys: and mount a backup volume in its place. Then use the BACKUP package to preserve the files on the disk. Use a similar procedure to restore any files to the disk. If you lose the entire file structure of the disk, including the system and/or the BACKUP package, you can restore the disk using a system recovery medium as described in Software Dispatch Article 16.1.2.

Note: If you ever patch or replace on the system disk any of the BACKUP or system modules that you copied in the above procedure, you should do so while the nonsystem disk is dismounted. After replacing the module(s), mount the nonsystem disk with the /PRIVATE switch and use PIP with the /UP (update in place) switch to change the file on the nonsystem disk. Repeat for each disk used on drive xxn: that contains a minimal system for backup purposes. Also, if any patches are installed to the RSTS/E monitor or the INIT.SYS code, be sure to install them on all disks which contain copies of a V06B system.

CREATING A SYSTEM DISK RECOVERY MEDIUM

Regular use of the RSTS/E BACKUP package (described in Chapter 18 of the RSTS/E System User's Guide and Chapter 8 of the System Manager's Guide) provides protection against loss of files if the system crashes, if disk errors occur, or if important files are accidently destroyed. However, the one situation from which BACKUP cannot recover is when the system disk or one or more of the monitor files is destroyed and the system cannot be started at all. One option in such an event is to use the RSTS/E distribution kit to regenerate the system, install all patches, build the complete system library, and then restore the remaining files on the system disk from the most recent backup. This is time consuming and error prone. Recovery is much simpler if you have a "recovery medium".

This article tells you how to create a system recovery medium (disk or tape) which contains your fully patched monitor, BASIC-PLUS Run-Time System, and enough of the BACKUP package to restore the rest of the system disk file structure. The recovery medium is bootable in a manner similar to the distribution medium and any intermediate tape (or disk) which you may have created during SYSGEN. In order to recover from destruction of the system disk, you will need the recovery medium and a recent backup of your system disk. Note that this backup must include an up-to-date version of the file [1,2]ACCT.SYS so that the recovery can enter all accounts properly. If you ever change your monitor, or install any patches in any of the files placed on the recovery medium, you should rebuild that medium.

To build the recovery medium, you must have ADDed the RT11 Run-Time System to RSTS/E. You must also have the program HOOK.SAV in account [1,2]. If either of these files is not already on your system disk, mount the first volume of your distribution kit (as device xxn:) write locked. (If your kit is on DECtape, mount the second volume to get HOOK.SAV.) Use the following commands to copy the files and add RT11 (computer output is underlined):

```
RUN $PIP
PIP     V06B-03 RSTS V06B-02 Your system
#SY:[0,1]RT11.RTS/CL:32/CO:T=xxn:[0,1]RT11.RTS
#SY:$HOOK.SAV/CO:T=xxn:$HOOK.SAV
#SY:$HOOK.SAV<124>/RE
#^Z

Ready

RUN $UTILTY
UTILTY  V06B-03 RSTS V06B-02 Your system
#ADD RT11
#NAME RT11=SY:$HOOK.SAV
#^Z

Ready
```

If you are creating a recovery magtape or DECtape, you will also need the source of extended PIP, PIPEXT.BAS. Copy this from your library distribution disk or tape. (If any patches have been installed in PIPEXT.BAS, use your saved copy of the modified source.) Now follow the instructions in one of the following sections, depending on whether you wish to build a recovery disk, magtape, or DECtape. In the examples, the file names RSTS.SIL and BASIC.RTS are used as the monitor SIL and default BASIC-PLUS Run-Time System. You should substitute the actual file names for your system.

Building a recovery disk

Using PIP or an editor, create the file RECOVR.CMD in [1,2] containing the following text:

```
SY0:$*.*/BL=IN:$BAC???.*
SY0:$*.BAC<232>/RE
SY0:$BACENT.BAC<104>/RE
```

Mount a formatted and initialized disk on a free disk drive. (The text below uses xxn: as the device designator and 'packid' as the pack identification for the recovery pack.) If necessary, shut down your system and use DSKINT to format a pack, or use the program DSKINT.BAC to initialize an already formatted scratch pack. Using UTILTY or UMOUNT, logically mount xxn:. Follow the procedure below to build the recovery disk (computer responses are underlined):

```
RUN $PIP
PIP    V06B-03 RSTS V06B-02 Your system
#xxn:[0,1]*.*/BL=SY:[0,1]INIT.SYS,RSTS.SIL,BASIC.RTS,ERR.ERR
#xxn:$*.*/BL=SY:$PIP.BAC,UTILTY.BAC
#xxn:$UTILTY.BAC<124>,PIP.BAC<104>/RE
#xxn:$*.*/BL=SY:$RECOVR.CMD,BACKUP.BAC,BACCON.BAC,BACLST.BAC,/MORE
MORE>   BACDIR.BAC,BACMNT.BAC,BACLAB.BAC,BACCOM.BAC,BACENT.BAC,/MORE
MORE>   BACLOD.BAC,BACFRM.BAC,BACKUP.PRM
#^Z

Ready

RUN $HOOK
*xxn:[0,1]INIT.SYS
Hook complete
*^Z

Ready
```

Dismount the recovery disk and save it. Also dismount the distribution medium.

To recover from the destruction of the system disk or one or more of the system files, you need the recovery disk and a recent BACKUP. Mount the recovery disk on one disk drive and bootstrap it. The RSTS/E initialization code runs and prints a header and the OPTION: prompt. On another disk drive, mount the disk on which to restore the system disk. Using the DSKINT option, recreate your system disk with the same parameters that you used at system generation time. Then use the COPY option to move your monitor files to the new system disk. As part of the COPY option, INIT bootstraps that disk. If necessary, use the HARDWR option to change controller characteristics. Then INSTALL your monitor SIL and use REFRESH to create the required system files on your system disk. There is no need to use DEFAULT or SET. START the system on your new system disk and continue as follows:

```
?Can't find file or account
?Program lost-sorry

Ready

ASSIGN xxn:IN
```

```
Ready

MOUNT IN:packid/RO

Ready

RUN IN:$PIP
PIP    V06B-03 RSTS V06B-02 Your system
#@IN:RECOVR.CMD
#^Z

Ready

RUN IN:$UTILTY
UTILTY  V06B-03 RSTS V06B-02 Your system
#DISMOUNT IN:
#^Z

Ready
```

Dismount the recovery disk from drive xxn:. Your new system disk now has enough of the BACKUP package to restore the entire disk from a recent BACKUP. Do this now, specifying that no files be superseded. You must now use the STANDARD option of REACT to enter all accounts that BACKUP did not enter. Ignore the error messages generated when REACT tries to enter accounts that already exist. Your system disk is now ready to use.

Building a recovery magtape

Using PIP or an editor, create the file RECOVR.CMD in [1,2] containing the following text:

```
SY0:$*.*/BL=IN:$*.*/RW:NO
SY0:$*.BAC<232>/RE
SY0:$BACENT.BAC<104>/RE
```

Mount a magtape on a free tape drive (referred to below as xxn:). Follow the procedure below to build the recovery tape (computer responses are underlined):

```
RUN $HOOK
*xxn:[0,1]INIT.SYS,SY:[0,1]INIT.SYS
Hook complete
*^Z

Ready

RUN $PIP
PIP    V06B-03 RSTS V06B-02 Your system
#xxn:[0,1]*.*/BL=SY:[0,1]RSTS.SIL,BASIC.RTS,ERR.ERR
#xxn:$*.*/BL=SY:$PIPEXT.BAS
#xxn:$*.*/BL=SY:$RECOVR.CMD,BACKUP.BAC,BACCON.BAC,BACLST.BAC,/MORE
MORE>   BACDIR.BAC,BACMNT.BAC,BACLAB.BAC,BACCOM.BAC,BACENT.BAC,/MORE
MORE>   BACLOD.BAC,BACFRM.BAC,BACKUP.PRM
#^Z

Ready
```

Dismount the tape and save it.

To recover from the destruction of the system disk or one or more of the system files, you need the recovery tape and a recent BACKUP. Mount the recovery tape on a tape drive (referred to below as xxn:) and bootstrap it. The RSTS/E initialization code runs and prints a header and the OPTION: prompt. Mount the disk on which to restore the system disk. Using the DSKINT option, recreate your system disk with the same parameters that you used at system generation time. Then use the COPY option to move your monitor files to the new system disk. As part of the COPY option, INIT bootstraps that disk. If necessary, use the HARDWR option to change controller characteristics. Then INSTALL your monitor SIL and use REFRESH to create the required system files on your system disk. There is no need to use DEFAULT or SET. START the system on your new system disk and continue as follows:

```
?Can't find file or account
?Program lost-sorry

Ready

ASSIGN xxn:IN

Ready
```

```
RUN IN:$PIPEXT
PIP     V06B-03 RSTS V06B-02 Your system
#$*.*/FA=IN:$RECOVR.CMD
#@RECOVR.CMD
#RECOVR.CMD/DE
#^Z

Ready
```

Dismount the recovery tape from drive xxn:. Your new system disk now has enough of the BACKUP package to restore the entire disk from a recent BACKUP. Do this now, specifying that no files be superseded. You must now use the STANDARD option of REACT to enter all accounts that BACKUP did not enter. Ignore the error messages generated when REACT tries to enter accounts that already exist. Your system disk is now ready to use.

Building recovery DECtapes

Recovery using DECtape requires two tapes.

Using PIP or an editor, create the file RECOVR.CMD in [1,2] containing the following text:

```
SY0:$*.*/CO:T=IN:$BAC???.*
SY0:$*.BAC<232>/RE
SY0:$BACENT.BAC<104>/RE
```

Mount two DECtapes on drives DT0: and DT1:. Follow the procedure below to build the recovery tapes (computer responses are underlined):

```
RUN $PIP
PIP      V06B-03 RSTS V06B-02 Your system
#DT0:/ZE
Really zero DT0: ? Y
DT0: zeroed
#DT1:/ZE
Really zero DT1: ? Y
DT1: zeroed
#DT0:*.*/CO=SY:[0,1]INIT.SYS,RSTS.SIL,BASIC.RTS,ERR.ERR
#DT1:*.*/FA=SY:$PIPEXT.BAS
#DT1:*.*/CO=SY:$RECOVR.CMD,BACKUP.BAC,BACCON.BAC,BACLST.BAC,/MORE
MORE>   BACDIR.BAC,BACMNT.BAC,BACLAB.BAC,BACCOM.BAC,BACENT.BAC,/MORE
MORE>   BACLOD.BAC,BACFRM.BAC,BACKUP.PRM
#^Z

Ready

RUN $HOOK
*DT0:INIT.SYS
Hook complete
*^Z

Ready
```

Dismount the tapes and save them.

To recover from the destruction of the system disk or one or more of the system files, you need the recovery tapes and a recent BACKUP. Mount the tapes on DECtape drives 0 and 1 (the way they were created) and bootstrap DT0:. The RSTS/E initialization code runs and prints a header and the OPTION: prompt. Mount the disk on which to restore the system disk. Using the DSKINT option, recreate your system disk with the same parameters that you used at system generation time. Then use the COPY option to move your monitor files to the new system disk. As part of the COPY option, INIT bootstraps that disk. If necessary, use the HARDWR option to change controller characteristics. Then INSTALL your monitor SIL and use REFRESH to create the required system files on your system disk. There is no need to use DEFAULT or SET. START the system on your new system disk and continue as follows:

```
?Can't find file or account
?Program lost-sorry
```

```
READY

ASSIGN DT1:IN

READY

RUN IN:$PIPEXT
PIP      V06B-03 RSTS V06B-02 Your system
#$*.*/FA=IN:$RECOVR.CMD
#@RECOVR.CMD
#RECOVR.CMD/DE
#^Z

Ready
```

Dismount the recovery tapes. Your new system disk now has enough of the BACKUP package to restore the entire disk from a recent BACKUP. Do this now, specifying that no files be superseded. You must now use the STANDARD option of REACT to enter all accounts that BACKUP did not enter. Ignore the error messages generated when REACT tries to enter accounts that already exist. Your system disk is now ready to use.

Notes on multiple disk systems

If your system disk is small, or if the public disk structure consists of more than one disk, some of the BACKUP or other programs may reside on a non-system public disk. In this case, you should copy to the recovery medium only those .BAC files that exist on the system disk. During the recovery procedure, after starting the system on your new system disk, you must physically mount and write-enable the other disks in your system's public structure. Type the command

```
MOUNT xxn:packid
```

for each non-system public disk, replacing xxn: with the disk name and unit, and packid with the pack ID for that disk. This procedure ensures that all required .BAC programs are available for the recovery.

Take the following precautions when creating a recovery disk:

1. If you can mount the recovery disk on an unused drive or in place of a private disk, do so. This leaves the full public disk structure mounted.

2. If you cannot mount the recovery disk in addition to the full public structure, you must dismount a non-system public disk. This disk must not contain any of the files required for the creation of the recovery disk or the actual recovery procedure. This disk can be dismounted while creating the recovery disk, and need not be mounted until a recovery is complete. If necessary, use PIP to copy files from this disk to another disk on the public structure:

   ```
   RUN $PIP
   PIP     V06B-03 RSTS V06B-02 Your system
   #out:TEMP=in:file.ext/BL
   #in:file.ext/DE
   #out:file.ext<prot>=out:TEMP/RE
   ```

   In these commands, in: is the name and unit number of the disk to be cleared of files, out: is the name and unit number of any other public disk, file.ext is a file to be moved, and prot is the protection code for the file (obtained from a directory of in:). Repeat these commands to PIP to copy each file on in: necessary for building the recovery disk or for the recovery procedure itself.

INSTALLING AUXILIARY RUN-TIME SYSTEM SUPPORT

You must install auxiliary Run-Time System support on a RSTS/E system in order to use one or more optional language processors such as BASIC-PLUS-2, COBOL, and FORTRAN. The language and version determine the Run-Time System required, as given in the table below:

Language (Version)	Run-Time System
BASIC-PLUS-2	RSX
COBOL-11 (V02)	RTSLIB
COBOL-11 (V03)	RSX
FORTRAN	RT11
RPG-II (V7.06)	RTSLIB
RPG-II (V8)	RSX

In addition, the Record Management Services (RMS) subsystem requires the RSX Run-Time System.

The following paragraphs give the procedures you must follow to add these Run-Time Systems to your RSTS/E system. You should also edit your START.CTL file (or some indirect command file invoked by START.CTL) to include commands to the UTILTY program which "ADD" these auxiliary Run-Time Systems. Refer to the System Manager's Guide, Section 7.1.4.1 for more information.

Installing the RT11 Run-Time System

The RT11 Run-Time System and its associated utilities are distributed on the RSTS/E V06B-02 System Generation kit. If you specified the same disk pack for your SYSGEN system and your target system (i.e. answered 'SY' to the 'Output to?' query), then the RT11 Run-Time System and most of the utilities are already on your disk. You should copy one other utility, PATCH.SAV, from your distribution kit before installing the FORTRAN language processor.

If you placed your target system on a different disk at SYSGEN time, you must copy the RT11 Run-Time System and the utilities from the distribution kit. Mount the first volume of your RSTS/E V06B-02 distribution kit on device xxn: and write lock it. This volume is labeled as follows:

Medium	Label
Magtape (7-track)	DEC-11-ORSPA-E-MC7
Magtape (9-track)	DEC-11-ORSPA-E-MC9
DECtape	DEC-11-ORSBA-E-UC1
RK05 cartridge	DEC-11-ORSPA-E-HC
RK06 cartridge	DEC-11-ORSPA-E-BC

To mount a distribution disk, use the pack ID 'ORSPAE' and include the '/RO' switch in the mount command. Log in under a privileged account and use the following commands to copy the RT11 Run-Time System and utilities. Your required input is underlined and all input is terminated with carriage return.

```
OPEN "SY0:[0,1]RT11.RTS/MO:16/SI:20" FOR OUTPUT AS FILE 1%

Ready

RUN $PIP
PIP     V06B-03 RSTS V06B-02 Your system
#SY0:[0,1]RT11.RTS/CO:T/UP=xxn:[0,1]RT11.RTS
#^Z

Ready
```

(Start here if you are copying only PATCH.SAV. Specify only that one input file in the command string to PIP.SAV which appears below.)

(If your distribution medium is DECtape, dismount DEC-11-ORSBA-E-UC1 from xxn: and mount the reel labeled DEC-11-ORSBA-E-UC2 on that unit.)

```
RUN $UTILTY
UTILTY  V06B-03 RSTS V06B-02 Your system
#ADD RT11
#^Z

Ready

A$=SYS(CHR$(9%)+MID(SYS(CHR$(6%)+CHR$(-10%)+'RT11'),7%,4%))

.SIZE 8

.R xxn:PIP.SAV
*$*.*<40>=xxn:$PIP.SAV,$MACRO.SAV,$CREF.SAV,$LINK.SAV,$LIBR.SAV,$PATCH.SAV/N
*^Z

.R LOGOUT

(continue with the normal LOGOUT procedure)
```

Installing the RTSLIB and RSX Run-Time Systems

Use the BUILD system program to install the RTSLIB and RSX Run-Time Systems and their associated utilities. Mount the appropriate volume of the system library from your RSTS/E V06B-02 distribution kit on device xxn: and write lock it. This volume is labeled as follows:

Medium	Label
Magtape (7-track)	DEC-11-ORS1A-E-MA7
Magtape (9-track)	DEC-11-ORS1A-E-MA9
DECtape	DEC-11-ORSCA-E-UA11
RK05 cartridge	DEC-11-ORSLA-E-HA1
RK06 cartridge	DEC-11-ORSLA-E-BA

To mount a distribution disk, use the pack ID 'ORSLAE' and include the '/RO' switch in the mount command. Log in under a privileged account and use the following commands to build the RTSLIB and RSX Run-Time Systems and their associated utilities. Your required input is underlined and all input is terminated with carriage return.

```
RUN $BUILD
BUILD   V06B-03 RSTS V06B-02 Your system
SYSTEM BUILD? NO
SOURCE INPUT DEVICE? xxn:
LIBRARY OUTPUT DEVICE <SY:>? <cr>                    (use the default)
LIBRARY ACCOUNT <[1,2]>? <cr>                        (use the default)
CONTROL FILE IS? $AUXBLD.CTL
^C
HELLO

RSTS V06B-02 Your system  Job 2  KB0  05-Jan-77  06:25 PM
# 1 / 2
Password:
Job 1 is detached under this account
Job number to attach to?
1 other user is logged in under this account


Ready

ASSIGN [1,2]

Ready
```

```
! ********** AUXBLD.CTL - SET UP THE AUXILIARY RUN-TIME SYSTEM

OLD xxn:$TKB

Ready

COMPILE SY :@TKB

Ready

OLD xxn:$LNGBLD

Ready

COMPILE SY :@LNGBLD

Ready

OLD xxn:$FLX

Ready

COMPILE SY :@FLX40>

Ready

RUN $PIP
PIP     V06B-03 RSTS V06B-02 Your system
#SY :[0,1]RTSLIB.RTS     =xxn:$RTSLIB.RTS/CO:T/CL:32
#SY :[0,1]RSX   .RTS     =xxn:$RSX   .RTS/CO:T/CL:16
#SY :[1,1]SYSLIB.OLB<40>=xxn:$SYSLIB.OLB/CO:T
#SY :[1,1]RSXMAC.SML<40>=xxn:$RSXMAC.SML/CO:T
#SY :[1,1]ODT   .OBJ<40>=xxn:$ODT   .OBJ/CO:T
#SY :@    BIGTKB.TSK     =xxn:$BIGTKB.TSK/CO:T
#SY :@    TKB   .TSK     =xxn:$TKB   .TSK/CO:T
#SY :@    MAC   .TSK     =xxn:$MAC   .TSK/CO:T
#SY :@    LBR   .TSK     =xxn:$LBR   .TSK/CO:T
#SY :@    FLX   .HLP<40>=xxn:$FLX   .HLP/FA
#SY :@    BIGTKB.TSK<104>/RE
#SY :@    TKB   .TSK<104>/RE
#SY :@    MAC   .TSK<104>/RE
#SY :@    LBR   .TSK<104>/RE
#^C

Ready
```

```
RUN $UTILTY
UTILTY  V06B-03 RSTS V06B-02 Your system
#NAME RTSLIB=SY :@BIGTKB.TSK
#NAME RSX   =SY :@TKB   .TSK
#NAME RSX   =SY :@MAC   .TSK
#NAME RSX   =SY :@LBR   .TSK
#EXIT

Ready

 C

Ready

HELLO

RSTS V06B-02 Your system  Job 2  [1,2]  KB0  05-Jan-77  06:29 PM
Job 1 is detached under this account
Job number to attach to?  1
Attaching to job 1

BUILD COMPLETE

Ready
```

If your distribution is on DECtape, you must build the RTSLIB and RSX Run-Time Systems in two stages. Perform the procedure shown above with DEC-11-ORSCA-E-UA11 mounted. Not all of the files shown in the example will be created. When the build is complete, dismount DEC-11-ORSCA-E-UA11, mount the tape labeled DEC-11-ORSCA-E-UA12, and repeat the procedure. The control file on that tape will finish building the required files.

FORTRAN IV/RSTS-E V01C for RSTS/E V06B-02
FORTRAN IV/PACKAGE NOTES

FORTRAN IV INSTALLATION PROCEDURE

This article describes the procedure for installing FORTRAN IV V01C on RSTS/E V06B-02 from the FORTRAN distribution kit.

NOTE

FORTRAN IV V01C has been re-released for RSTS/E V06B-02. If FORTRAN has been previously installed on your RSTS/E system under V06A-02, you must obtain a new FORTRAN distribution kit and re-install FORTRAN.

Before installing FORTRAN, ensure that the following requirements are satisfied:

1. The Run-Time System RT11 must be on the system and added as an auxiliary Run-Time System (see Article 20.1.1).

2. The default swap maximum may require change for proper FORTRAN operation. The FORTRAN installation procedures require a swap maximum of at least 12K. The installation will not proceed properly if this requirement is not met. A large swap maximum (up to 28K) can be used if operation of large FORTRAN programs is anticipated. Refer to the System Generation Manual for information on the procedures for modifying the swap maximum at system startup. If the system as currently set up does not allow the swap maximum desired for FORTRAN operation (due to insufficient swap file space, etc.), use the Initialization option REFRESH to create more swap space and DEFAULT to increase the swap maximum.

3. The FORTRAN installation procedure for magtape distribution medium requires that the MAGTAPE LABELLING DEFAULT be set to DOS. If the standard labelling default for the operating system is ANSI, use the DEFAULT option of the Initialization options to reset the labelling default to DOS before proceeding with the FORTRAN installation. The default can be restored to ANSI after FORTRAN is installed.

The following procedure describes how FORTRAN is installed from a magtape distribution medium. The procedure for building from a DECtape or disk cartridge medium is similar. The order numbers for the respective media are listed below:

Medium	Order Number
Magtape (7-track)	DEC-11-LRFSA-B-MC7
Magtape (9-track)	DEC-11-LRFSA-B-MC9
DECtape	DEC-11-LRFSA-B-UC
Disk cartridge (RK05)	DEC-11-LRFSA-B-HC
Disk cartridge (RK06)	DEC-11-LRFSA-B-BC

If the distribution medium is magtape, ensure that there is no write enable ring on the reel. Mount the distribution medium on an available drive and write lock the drive (if DECtape) or write protect it (if disk cartridge). Ready the drive by positioning the magtape at load point and setting the drive to ON-LINE, or by setting the DECtape drive to REMOTE, or by setting the disk drive to the RUN condition.

NOTE

If the distribution medium is a disk, it must be logically mounted using the MOUNT command, using a pack label of "FORTRA", e.g.

```
MOUNT DK0:FORTRA/RO
```

Log into any privileged account and run the BUILDR program from the distribution medium, as shown in the following example. All user supplied input is underlined and should be terminated with carriage return.

```
NEW RT11.RTS

.R MT0:BUILDR
DETACHING...
HELLO 1,xxx
Password:
Job 1 is detached under this account
Job number to attach to?
1 other user is logged in under this account

Ready

A$=SYS(CHR$(9%)+MID(SYS(CHR$(6%)+CHR$(-10%)+'RT11'),7%,4%))

.ASSIGN  MT0: INPUT

.R PIP.SAV
*SY:=INPUT:INSTAL.CTL
*^Z

.R PIP.SAV
*$=INPUT:$*.OBJ,$OTI.*,$FORTRA.SAV,$FORTRA.HLP/N
*$FORTRA.SAV<104>/R
*$FORTRA.HLP<40>/R
*SY:=INPUT:$DEMO.FOR,$FORCFG.SAV
*^Z
```

```
.RN FORCFG.SAV
---------- Configuration Questions ----------

Answer questions with:
<RETURN> for more information, YES(Y) for affirmation
or anything else for NO.

Default sizes for RSTS FORTRAN components are:
      FORTRAN compiler         8K words
      LINK                    28K words
      MACRO assembler         28K words

Do you wish to change the defaults ?
```

If the defaults as specified above are satisfactory for the installation, reply NO to the question. If the answer YES or Y is given, BUILDR will then ask, in turn, whether the FORTRAN compiler, the Linker, and the MACRO assembler are to be changed in size. Answer YES or NO to each question as appropriate. If the answer YES is given, BUILDR will ask for the new size. Reply with the desired size as a number in the range 6 to 28.

```
Do you wish to change the defaults ?Y
Do you wish to change FORTRAN ?Y
      Enter new size in K words: 16
Do you wish to change LINK ?N
Do you wish to change MACRO ?N
```

```
        --FORTRAN Library Selection --

        The system default library, $FORLIB.OBJ, is being
        built for FPU hardware.

        Will stand-alone FORTRAN support (to run programs
        on satellite machines) be used on this system?
```

Several prototype FORTRAN library components are included on the distribution medium. They differ in their use of processor hardware (FIS, EAE, etc.). The proper prototype component has been selected for your particular hardware configuration and will be used to build the default FORTRAN library, $FORLIB.OBJ.

If required, the installation procedure will build FORTRAN libaries for other hardware configurations. These extra libraries are usually used to create FORTRAN programs on a host machine which can be run stand-alone on satellite machines having different hardware requirements. If other libraries are not required, answer NO to the question. If additional libraries are required, reply YES to the question. The question "IS THE XXX LIBRARY REQUIRED ?" will then appear for each hardware option (other than that which is present on the host system). Reply YES to each question concerning a particular library which should be generated, or NO if that library should not be built.

```
        Will stand-alone FORTRAN support (to run programs
        on satellite machines) be used on this system?Y
        Is the FIS library required ?N
        Is the EIS library required ?Y
        Is the EAE library required ?N
        Is the NHD library required ?N

        ------ Configuration Dialogue Complete ------
```

```
.R PATCH.SAV

PATCH V01-02

FILE NAME--
*$FORTRA/O
*56/  0       20
*E
..R PATCH.SAV

PATCH V01-02

FILE NAME--
*$LINK/O
*56/  34      34
*E
..R PATCH.SAV

PATCH V01-02

FILE NAME--
*$MACRO
*56/  34      34
*E
..R LIBR.SAV
*$FORLIB=UNI,OTSCOM,FOPEN,V2NS,FPU/G

ENTRY POINT:
$ERRS
$ERRTB

$FIO2 ILL INS
$END2 ILL INS
$END2 ILL INS
$CLO2 ILL INS
$FIO2 ILL INS
$FIO2 ILL INS
```

```
*$FORLIB<40>/CL:32.=$FORLIB,OTI.FPU/U/G

ENTRY POINT:
$ERRS
$ERRTB

$FIO2 ILL INS
$END2 ILL INS
$END2 ILL INS
$CLO2 ILL INS
$FIO2 ILL INS
$FIO2 ILL INS

*$EISLIB<40>=UNI,OTSCOM,FOPEN,V2NS,EIS/G

ENTRY POINT:
$ERRS
$ERRTB

$FIO2 ILL INS
$END2 ILL INS
$END2 ILL INS
$CLO2 ILL INS
$FIO2 ILL INS
$FIO2 ILL INS

*^Z
```

FORTRAN IV/RSTS-E V01C for RSTS/E V06B-02 Seq 23.1.1 N
FORTRAN IV/PACKAGE NOTES 7 of 8

When all components of the FORTRAN system are complete, a demonstration test of the FORTRAN system is run. This demonstration test compiles, links, and executes a program to ensure that the installation procedure has been successful. The test output is self-explanatory.

```
.R FORTRA.SAV
*DEMO,KB:=DEMO

FORTRAN IV    V01C-03F        THU 06-JAN-77 17:47:48          PAGE 001

      C
      C       DEMONSTRATION TEST PROGRAM FOR RSTS FORTRAN IV
      C
0001          TYPE 100
0002  100     FORMAT(' ***** RSTS FORTRAN IV DEMONSTRATION TEST *****')
0003          A = 3.14159
0004          A = A / 3.14159
0005          IF (A.NE.1.0) TYPE 200
0007  200     FORMAT('0ERROR IN FLOATING POINT ARITHMETIC TEST.')
0008          I = A
0009          I = I*8 / 4
0010          IF (I.NE.2) TYPE 300
0012  300     FORMAT('0ERROR IN INTEGER ARITHMETIC TEST.')
0013          TYPE 400
0014  400     FORMAT('0INSTALLATION SUCCESSFUL IF NO ERROR MESSAGES',/
             +    '              WERE PRINTED ABOVE.')
0015          TYPE 500
0016  500     FORMAT('0**** FORTRAN DEMONSTRATION TEST COMPLETE *****')
0017          CALL EXIT
0018          END

FORTRAN IV    STORAGE MAP

NAME  OFFSET  ATTRIBUTES

A     000426  REAL*4    VARIABLE
I     000432  INTEGER*2 VARIABLE
EXIT  000000  REAL*4    PROCEDURE
*^Z

.R LINK.SAV
*DEMO,DEMO=DEMO/F

*^Z

.RN DEMO.SAV

***** RSTS FORTRAN IV DEMONSTRATION TEST *****

INSTALLATION SUCCESSFUL IF NO ERROR MESSAGES
          WERE PRINTED ABOVE.

**** FORTRAN DEMONSTRATION TEST COMPLETE *****
```

```
.R LOGOUT
Confirm: Y
Saved all disk files; xxx blocks in use
Job 2 User 1,xxx logged off KB0 at 01-Jan-77 05:48 PM
1 other user still logged in under this account
System RSTS V06B-02 Timesharing
Run time was 27.6 seconds
Elapsed time was 8 minutes, 54 seconds
Good afternoon


BUILDR COMPLETE

Ready
```

The FORTRAN installation procedure is now complete, and the distribution medium may be removed from the drive, after first logically dismounting it if it is a disk.

CLARIFICATION: COMPARING ASCII DATA ITEMS (SPR 11-8572 RB)

When more than one character is stored in an individual FORTRAN variable (as is done with A-FORMAT conversion), comparison between variables may not yield the expected ASCII collating sequence. For example, in the following FORTRAN code, the variable LESS is set to .TRUE.:

```
LOGICAL LESS
DATA A/'SF'/, B/'DP'/, LESS/.FALSE./
IF (A.LT.B) LESS = .TRUE.
```

The FORTRAN language does not specify the number of character data which may be stored in a data item; it merely guarantees that one character (minimum) can be accommodated in each variable. The PDP-11 format for ASCII data stores succeeding characters at ascending byte addresses; however, integer and floating-point comparison operations compare word, not byte, data, hence the items are compared byte-reversed from the ASCII sense. To correctly compare ASCII data in the PDP-11 environment, the LOGICAL*1 datatype should be used (which will cause byte comparisons to take place), or the program should be restricted, as per the FORTRAN standard, to storing only one ASCII character per variable or array element; e.g., A1 FORMAT.

UNLOCKING FILES OPENED FOR SHARED ACCESS (GH)

When opening a single file for shared access it is often necessary to unlock the file after reading a record (if that record is not to be modified) so that other programs may access it.

The following FORTRAN callable subroutine will unlock the file on a given FORTRAN LUN. A typical use would be as follows:

```
      DIMENSION ARR(256)
      LOGICAL*1 IERR
      DEFINE FILE 1(10,256,U,IU)
      CALL  OPEN(1,'TEST.DAT',,'OLD','NC',1,'RAN',,,,1)

            .
            .
            .

      READ(1'IREC)ARR
      CALL UNLOCK(1,IERR)
      IF(IERR.NE.0)TYPE 100,IERR
100   FORMAT(' UNLOCK ERROR #',I3)
```

```
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        .TITLE  UNLOCK ROUTINE FOR RSTS/E FORTRAN IV.

        .GLOBL  $FCHNL,UNLOCK,$AOTS

;
;       FORTRAN CALL:
;
;           CALL UNLOCK(LUN,IOSTATUS)
;
;           WHERE:
;
;               LUN = FORTRAN LUN #.
;               IOSTATUS = IOSTS AS RETURNED FROM RSTS/E.
;
;               NOTE:   LUN IS INTEGER*2, IOSTATUS IS LOGICAL*1
;
```

```
R0      = %0
R2      = %2
R3      = %3
R5      = %5
PC      = %7
.PRIV   = 104377
.SPEC   = 104014
DSKHND  = 0
XRB     = 442
XRLEN   = 0
XRCI    = 6
IOSTS   = 402
CHNLNO  = 17.

UNLOCK: MOV     @#$AOTS,R3
        MOV     @2(R5),R2
        JSR     PC,$FCHNL
        MOVB    CHNLNO(R0),R0
        ASL     R0
        ADD     @#54,R0
        MOV     142(R0),R0
        MOV     14(R0),XRB+XRCI
        CLR     XRB+XRLEN
        MOVB    #DSKHND,XRB+XRCI+1
        .PRIV
        .SPEC
        MOVB    @#IOSTS,@4(R5)
        RTS     PC
        .END
```

ENTERING LOWER CASE FROM TERMINAL (GH)

Input entered from the terminal is changed to upper case even though TTYSET is used to set LC input.

The following FORTRAN callable subroutine should be called whenever lower case input is desired from the terminal. Note that TTYSET must still be used to set LC input.

A typical use follows:

```
                LOGICAL*1 L(80)
                CALL SETLC
        1       ACCEPT 100,L
                TYPE 200,L
                GO TO 1
        100     FORMAT(80A1)
        200     FORMAT(1X,80A1)
                END
```

```
==========================================================================

        .TITLE  SET LC INPUT ROUTINE
        .GLOBL  SETLC
SETLC:  BIS     #40000,@#44
        RTS     %7
        .END
```

RSTS/E FORTRAN IV V01C for RSTS/E V06B-02 — Seq 23.2.1
also for V06A-02 — Seq 14.1.23
COMPILER V01C-03 F — 1 of 3

CERTAIN "ENCODE/DECODE" STATEMENTS ARE FLAGGED AS SYNTAX ERRORS.

Certain ENCODE/DECODE statements are flagged as syntax errors. The following patch corrects this problem.

1. Log onto account [1,2], the system library account.

1a. Verify that the current version of the compiler is V01C-03F. This may be done as follows:

```
 RUN $FORTRAN
*,TT:=TT:
^Z          (type CTRL Z here)

FORTRAN IV  V01C-03F
```

If the current version is not V01C-03F, then previous patches may have been omitted and should be applied before proceeding any futher.

2. Create the following file:

File name: TEST.FOR

```
      INTEGER*2 I2(2)
      ENCODE(2,100,I2   )J
      ENCODE(2,100,J    )J
      ENCODE(2,100,I2(I))J
      ENCODE(2,100,I2   ,ERR=200)J
      ENCODE(2,100,J    ,ERR=200)J
      ENCODE(2,100,I2(I),ERR=200)J
100   FORMAT(1X)
200   END
```

3. Create a copy of the compiler. This is the copy to be patched.

```
 RUN $PIP
*NEWFOR.SAV=FORTRA.SAV/CO
*NEWFOR.SAV<104>/RE
*^C
```

4. In the following the user types the underscored text; <CR> denotes the carriage return key; <LF> denotes the line feed key. The resulting version of FORTRAN will be FORTRAN IV V01C-03G.

```
 RUN $PATCH
 -------
PATCH V01-02

FILE NAME--
*NEWFOR/O<CR>
 ------------
```

```
*2210;1R
 -------
*2260;2R
 -------
*7:1,26/          112724   4737<LF>
 -------                   --------
7:1,30/ 26        3660<LF>
                  --------
-------------------------------------------------------------------
7:1,32/ 112724    <LF>
                  ----
7:1,34/ 106       26<CR>
                  ------
*7:2,472/         4767     4737<LF>
 --------                  --------
7:2,474/          702      2236<CR>
                           --------
*7:2,520/         4737     112724<LF>
 --------                  ----------
7:2,522/          2236     106<CR>
                           -------
*1:2,2506\        106      107<CR>
 ---------                 -------
*E
 -
```

5. Test the patch by compiling the FORTRAN test program, directing the listing to a file as follows:

```
 RUN $EXEC
*NEWFOR
*,TEST=TEST
*↑C             (CTRL C is typed here)
```

6. If the compiler aborts, loops, or the generated listing is not the same as the one below, reapply the patch starting at step 3 (above).

7. Once the compiler has been successfully patched, replace the original with the patched version as outlined below:

   Users should refrain from invoking FORTRAN for the duration of this step; the message

   Can't find file or account at line 410

   will be generated if the file $FORTRA.SAV is not present when either the RUN $FORTRA or the 'FOR' CCL is used to invoke FORTRAN.

```
e.g.          RUN $PIP
              #FORTRA.SAV/DE
              #FORTRA.SAV/RE=NEWFOR.SAV
              #↑C
```

8. Now that the patch is installed, the patched version of FORTRAN should be copied onto some backup medium for possible future patches.

```
e.g.          RUN $PIP
              #xx:FORTRA.SAV=FORTRA.SAV/CO    xx is the backup device.
              #↑C
```

RSTS/E FORTRAN IV V01C for RSTS/E V06B-02
also for V06A-02
COMPILER V01C-03 F

LISTINGS

```
FORTRAN IV      V01C-03G

0001          INTEGER*2 I2(2)
0002          ENCODE(2,100,I2   )J
0003          ENCODE(2,100,J    )J
0004          ENCODE(2,100,I2(I))J
0005          ENCODE(2,100,I2   ,ERR=200)J
0006          ENCODE(2,100,J    ,ERR=200)J
0007          ENCODE(2,100,I2(I),ERR=200)J
0008   100    FORMAT(1X)
0009   200    END


FORTRAN IV      STORAGE MAP

NAME     OFFSET   ATTRIBUTES

I2       000006   INTEGER*2 ARRAY (2)
J        000016   INTEGER*2 VARIABLE
I        000020   INTEGER*2 VARIABLE
```

OUTPUTTING REAL NUMBERS (SPR 11-10286 GH)

When ENCODING a floating point number whose value is between Ø and 1 via an F format specification with Ø places to the right of the decimal point; e.g., F5.Ø, the leading (significant) zeroes to the right of the decimal point may be ENCODED. The decimal point, however, will be aligned. This problem only occurs when the number of places to the right of the decimal point is specified to be Ø.

For example, the value Ø.ØØ5 ENCODED via F3.Ø format specification may be ENCODED as ⱭØ.ØØ.

This problem is due to a complex software problem in the format conversion routines. It will be considered in any future release. This problem may be avoided by specifying at least one place to the right of the decimal point when outputting or ENCODEing data.

END-OF-FILE NOT RECOGNIZED FROM KEYBOARD (SPR 11-9224 GH)

FORTRAN does not recognize control Z as end of file from the keyboard.

When the FORTRAN runtime system receives a control Z in response to a .TTYIN EMT, it exits the runtime system. When a control Z is received in response to a .READ EMT, an end of file is passes back to the program. The FORTRAN object time system does .TTYIN EMT requests for all teletype input. Hence, a control Z in response to a FORTRAN program expecting input from the terminal will terminate the program and exit the RTS.

RSTS/E FORTRAN IV V01C for RSTS/E V06B-02 Seq 23.4.1
also for RSTS/E V06A-02 Seq 14.1.13
RUNTIME SYSTEM 1 of 1

FILE STRUCTURES (SPR 11-10867 GH)

DSM-500 ISAM/RAM file structures cannot be copied or modified using PIP or other standard utilities.

Since all of the characteristics of a DMS-500 ISAM/RAM file structure are actually contained in the file itself (see FCB description), moving or changing its name will destroy its validity. DMSUTL only must be used to copy, rename or delete ISAM/RAM file structures.

UNEXPECTED ERROR ENCOUNTER (SPR 11-10530 GH)

DMS-500 ISAM/RAM functions sometimes return status code values greater than 256.

If the DMS-500 ISAM/RAM functions encounter a CTS-500/E or RSTS/E error that is unexpected, they return a status code with a value of the ERR value plus 256; e.g.:

FNS%=267%, means, ERR=11%

EXTENDING EXISTING FILES (GH)

EXTENDing an existing DMS-500 file can cause the file to become corrupted.

In the module DMSFSX.BAS, either insert (or replace) the following lines, or make the underlined corrections using an editor.

```
21000!  DMS-500          VERSION 2        EDIT 03
!!!!!!
!!!!!!  Copyright (C) 1975,1976,
!!!!!!  Digital Equipment Corporation, Maynard, Mass.
!!!!!!
21595   FIELD #(Q%+Q1%),Q9%*Q%(Q%,7%) AS Q0$,508%-LEN(Q0$) AS Q0$,1% AS Q4$ \
        LSET Q4$=CHR$(Q9%) IF Q%(Q%,12%)
21597   Q9%=Q%(Q%,5%) IF Q9%<Q%(Q%,5%) \ Q%(Q%,4%)=Q9% \
        IF CVT$%(Q3$)=0% THEN LSET Q2$=CHR$(Q%(Q%,2%)) \
                LSET Q3$=CVT%$(SWAP%(Q%(Q%,3%)+1%))
21600   LSET Q0$="" \ IF Q%(Q%,12%)=0% THEN Q%(Q%,3%)=Q3% ELSE
        IF Q1%=0% THEN PUT #Q%, RECORD Q3% \
                                FIELD #Q%, 512% AS Q0$ \ LSET Q0$=Q8$
```

STRTAD ENTRY IN FDB IS NOT ALWAYS RESET (SPR 11-8871 FH)

When using the create function of the IAMGEN program, if a file is used for the FDB source and no input data file is used, then the STRTAC (third) entry in the File Descriptor Block is not reset.

In line 95 of the IAMGEN program, include the variable F%(N3%) after F%(N4%) so that they are both set equal to N1%. The line should then read as follows:

```
95      F%(N5%)=N2%: F%(N4%),F%(N3%)=N1%:IF I6$="" THEN I0,I1%=0. ELSE
        OPEN I6$ FOR INPUT AS FILE N1%: GET #N1%:
        FIELD #N1%,N2% AS I0$,N2% AS I1$,N2% AS I2$:
        I0=FNF(CVT$%(I0$)): I1%=CVT$%(I1$): I2%=CVT$%(I2$)
        !GRAB HEADER
```

Note that this patch need not be made if only the IAM functions and programs are used to access IAM files, since they do not reference that entry.

REVISED BLANK RECORD HANDLING (SPR 11-9825 FH)

Blank records cause the following problems:

1. The REORG function of IAMGEN treats blank records as active records and does not count them as expansion slots nor provide an index to them.

2. The CREATE function of IAMGEN accepts blank records into the DRF but does not index to them.

3. There is no way to keep blank records out of the DRF at creation time or to eliminate them later using REORG or INDEX. Even though no indices are ever created to these records, they still occupy file space.

The following patch will cause both the CREATE and REORG functions to ignore blank records and return the records to expansion space.

Create or modify line numbers 2ØØ4Ø, 2ØØ7Ø, and 2Ø14Ø to read as follows:

```
20040   U6$=MID(U5$,F%(14%)-N1%,F%(12%)) :
        U6%=F%(N4%) : U7%=F%(N5%)
        !U6$=KEY,U6%=BLK,U7%=REC#

20070   IF U6$=" " AND ( C6%=N1% OR C6%=N2% ) THEN 20140 ELSE
        F%(N5%)=F%(N5%)+N1% : F%(N6%)=F%(N6%)+N1%
        ! UPDATE # RECS ADDED IF NOT BLANK OR IF BLANK AND INDEX FUNC

20140   IF U6$=" " THEN FNX9%=N6% : GOTO 20330
        !RETURN WITH UPDATED POINTERS ON BLANK KEY AND INDEX FUNCTION
```

Update the header to read Version 02, EDIT 02.

IMPROPERLY HANDLED LOGICAL DEVICE NAMES (SPR 11-10295 FH)

DSORT does not properly handle logical device names, causing the package to fail.

Modify line numbers 2Ø22Ø and 2Ø278 to read as follows:

```
20220   P6$=LEFT(F$,INSTR(1%,F$,":")-1%) :
        F$=SYS(CHR$(6%)+CHR$(-10%)+F$)
        ! SAVE THE DEVICE AND DO IT TO IT
20278   P4$=P6$ IF (F% AND 16384%) : P4$=FNR$(P4$,4%)
        ! CHANGE P4$ TO THE LOGICAL NAME IF ANY : TRANSLATE TO RAD50
```

Update the headers to read EDIT 02.

MINIMUM RECORD LENGTH FOR "DSORT" (SPR 11-10697 FH)

Minimum record length for DSORT, documented to be 6 bytes, is implemented as 16.

Alter line 11Ø3Ø in DSORT to read:

... IF D9%<6% OR ...

Update the headers to read REV:Ø3.

DMS-500 files still have a minimum of 16 but the value input is not referenced when processing them.

INDIRECT COMMAND FILE CREATION (SPR 11-10697 FH)

The /IC and /C switch options fail when attempting to enter the number of records in a Type 1 file without header or a Type 2 file.

In DSORT, alter or create line numbers 9162, 16062, and 16064 to correct this problem. Those lines should read as follows:

```
9162    IF ERL=16064 THEN IF ERR<>11% THEN RESUME 16064 ELSE
        CLOSE 3 : RESUME 1000
        ! IF HE TYPES A CTRL/Z BAG IT ELSE REPROMPT

16062   PRINT#1,"*"; : INPUT LINE#4,I$ : PRINT#3,I$; :
        IF I$=CHR$(13%)+CHR$(10%) THEN 16064 ELSE 16062
        ! NOTICE HOW WE GIVE THE RIGHT PROMPT AND THEN TAKE
          ANYTHING FOR KEYS

16064   IF ASCII(RIGHT(I3$,3%))>47% THEN 16020 ELSE
        PRINT#1,"RECORDS: "; : INPUT LINE#4,I$ : PRINT#3,I$; :
        GOTO 16020
        ! IF RECORD LENGTH STARTS WITH A DIGIT THEN TYPE 1 W/HEAD ELSE
          ASK FOR THE NUMBER OF RECORDS
```

Update the headers to read REV.Ø4.

DSORT TEST FOR DMS-500 FILE ERRONEOUS (FH)

There is a typographical error at line 2Ø47Ø of both DSORTD and DSORTF which causes the test for DMS-500 file structure to be false. This causes already existing DMS-500 output structures to be overwritten.

Line 2Ø47Ø should read:

```
2Ø47Ø     FND5%=-1% : GOTO 2Ø49Ø
```

The underline indicates the correction. It should be made to both DSORTD and DSORTF; and they both should be changed to indicate V08-Rev 02.

DMS-500 V02 for RSTS/E V06B-02 Seq 24.20.1*
for RSTS/E V06A-02 Seq 15.20.1*
DSORTD V08-Rev 01 1 of 1

DSORTD MAY RETURN A FATAL ERROR USING MULTIPLE KEY FILES (FH)

DSORTD under certain circumstances may return fatal errors while sorting multiple key files. (The files are unaffected.)

Change line 5230 of DSORTD to read:

```
523Ø IF K%=Ø% THEN
     IF F5% AND (K5%<>B%) THEN 694Ø ELSE K5%=B%
```

And change line 699Ø of DSORTD to read:

```
699Ø I$=STRING$(2%,Ø%)+CVT%$(P%(Ø%)):
     B$=CHR$(C1%)+CHR$(E%+4Ø%)+CHR$(H%):
     GOTO 569Ø
```

Update the header to indicate V08-Rev 03.

PREALLOCATION ERROR (SPR 11-10935 FH)

DSORTD will fail if it is unable to preallocate the number of blocks it needs for a keyfile.

Line 9Ø1Ø in DSORTD should read:

```
9Ø1Ø     if ERL=125Ø then ...
```

This correction will cause the error to be trapped, and as many blocks as possible allocated.

When DSORTD attempts to go beyond this during processing, it will attempt to open a second (or more) key file. If that attempt fails, then a fatal error will be reported.

FIELD OVERFLOW ERROR USING /F (FH)

When sorting from one DMS-500 file structure into another, if the DMS-500 ISAM key does not start at the same byte or if it is shorter and duplicate keys are not allowed, then DSORTF fails with a FIELD OVERFLOWS BUFFER AT LINE 21635 error.

Altering line number 7170 of DSORTF to the following, will correct the problem.

```
7170  IF K1%=P2%(4%)AND K2%>=P2%(3%)THEN 7174 ELSE
      P2%(4%)=K1%:P2%(3%)=K2%:D3%=-1%:
      P2%(9%)=5Ø8%/(K2%+3%)
      !IF SORT KEY POSITION <> FCB OR LENGTH<FCB THEN CHANGE THEM
      AND MAKE SURE WE CAN HAVE DUPLICATE KEYS
```

The alterations are underlined. Update the headers to read EDIT Ø4.

DSORT PRODUCES MISORDERED OUTPUT FILE WITH CORRECT KEYFILE (FH)

Under certain circumstances a misordered output file will be produced from a correctly ordered keyfile.

Add the following new line number to the DSORTX module:

```
3ØØ12   V1$=V1$+FNF$(1.*B%)
        ! SAVE B% FOR LATER
```

Update the edit level to Ø2 in the module headers. No other modules are affected; only the output files would have been misordered.

MISORDERED OUTPUT FILE ADDENDUM (SPR 11-9940 FH)

Under certain circumstances, a misordered output file may be produced from a correct key file.

Before installing the following patch, install the patch in sequence 15.22.1 dated September 1976.

Add line number 29Ø74 to DSORTS as follows:

```
29Ø74   N%=1Ø7% : B$=FNC4$
        ! SAVE B%
```

and modify line number 29Ø2Ø so that the assignment statement for V2$ reads as follows:

```
V2$=SYS(CHR$(8%)+V1$+STRING$(42%,Ø%)+B$)
```

Update the headers to read EDIT 02.

RSTS/E 2780 Package for RSTS/E V06B-02
Package Notes

2780 PACKAGE NOT YET AVAILABLE FOR RSTS/E V06B-02

The 2780 Package available for RSTS/E V06A-02 can not be used with V06B-02. Work on the V06B-02 2780 Package is progressing and the package will be released as soon as possible.

During a V06B-02 system generation, the 2780 package can be loaded from any distribution medium. There is no longer any requirement that the 2780 Package be available on the same distribution medium as the RSTS/E system.

When available from DIGITAL's Software Distribution Center, the RSTS/2780 Package which is compatible with RSTS/E V06B-02 will be identified by the following numbers.

DEC-11-ORCOA-D-UC	DECtape
DEC-11-ORCOA-D-MC9	9-track magtape
DEC-11-ORCOA-D-MC7	7-track magtape
DEC-11-ORCOA-D-HC	RK05 disk cartridge
DEC-11-ORCOA-D-BC	RK06 disk cartridge

COBRG TOO BIG FOR STANDARD COBOL COMPILER (SPR 11-8452 BR)

COBRG is too big for standard COBOL compiler.

Build 28K COBOL compiler using LNGBLD. When asked ANY CHANGES?, answer YES and specify a new size of 28.

CROSS REFERENCE PROBLEM (SPR 11-8522 FI)

Cross Reference listing option (/CREF) does not always work on RSTS/E.

The problem is due to faulty assignment of logical units to CREF scratch files in the task build command file (CBRSTS.CMD) for COBOL on RSTS/E. The list of logical unit assignments should be as shown below. Modify CBRSTS.CMD and rebuild COBOL-11 and CREF should work.

```
;CBRSTS.CMD
;
;THIS FILE IS THE COMMAND FILE FOR TKB TO BUILD THE COBOL-11 OVERLAY STRUCTURE
;FOR RSTS/E V.6 COBOL. THE OVERLAY DESCRIPTOR FILE IS CBRSTS.ODL
;
COBOL,COBOL=CBRSTS.ODL/MP
TASK = ...CBL                   ;TASK NAME
STACK = 512                     ;THE DEFAULT
UNITS = 14
;
;THE FOLLOWING TWO COMMANDS DEFINE THE INTERNAL STORAGE USED BY THE
;COMPILER (WFBLK) AND RUN-TIME SYSTEM (WFEND) FOR PROCESSING
;THE SOFTWARE VIRTUAL MEMORY WORKFILE.
;THE MINIMUM VALUES FOR THES AREAS ARE
;            WFBLK:6000 (OCTAL)
;      WFEND:30000 (OCTAL)
;THE NET EFFECT OF EXTENDING THE SIZE OF THESE AREAS IS TO SPEED UP THE
;COMPILER AND RUN TIME SYSTEM
;IN ADDITION THE AREA WFEND MAY HAVE TO BE EXPANDED DEPENDING ON THE
;SIZE OF THE COBOL PROGRAM OR THE NUMBER OF SIMULTANEOUSLY OPEN
;FILES IN THE RUNNING PROGRAM.
;******
;EXTENSION OF THESE AREAS SHOULD BE IN BLOCKS OF 2000 (OCTAL)
;*****
;
;IN GENERAL, WFEND MAY BE AS MUCH
;AS 22000 MORE THAN WFBLK WITHOUT PENALIZING TOTAL MEMORY USED,
;AS THEY ARE IN DIFFERENT OVERLAYS OF VASTLY DIFFERENT SIZE.
;THE MINIMUM VALUE WFBLK=6000,WFEND=30000 YIELDS A COBOL WHICH WILL
;RUN (ALBEIT SLOWLY) IN 18K OF USER MEMORY WITH ABOUT 10K CHARS OF
;RUN-TIME DATA AREA AND A FEW FILE BUFFERS.
;THESE PARAMETERS ARE ALTERED IN RSTS BY CBLBLD PROGRAM.
;
```

```
EXTSCT = WFBLK:32000
EXTSCT = WFEND:71000
EXTSCT = $$FSR1:6140
ASG = KB:1,KB:2,SY:3,KB:4,KB:5,SY:6,SY:7,SY:8,SY:9,SY:10,SY:11
ASG = SY:12,SY:13,SY:14
;
;CALL PARAMETERS FOLLOW
GBLDEF = CATBSZ:0
;
GBLDEF = NEWMAG:1                ;NEW MAG TAPE ENABLED
GBLDEF = PREFIX:000000           ;NULL/NULL
GBLDEF = POSFIX:005015           ;CR/LF
GBLDEF = CBLVER:31060            ;ASCII "02"
GBLDEF = CBLEDI:20101            ;ASCII "A "
GBLDEF = BUFCNT:1                ;1 FOR  SINGLE 2 FOR DOUBLE BUFFERING
GBLDEF = RSTSSW:1                ;NON-0 FOR RSTS
;
;LOGICAL UNIT ASSIGNMENTS (ABOVE)
;3 - WORKFILE
;4 - ACCEPT DEVICE LUN
;5 - DISPLAY DEVICE LUN
;6,7,8 - CREF SCRATCH FILES
;6 - SOURCE FILE INPUT AND FIRST USER FILE LUN
;7 - LIBRARY FILE INPUT
;8 - OBJECT PROGRAM FILE OUTPUT
;9 - INTERMEDIATE OBJECT FILE
;10 - LISTING FILE OUTPUT
;11 - CREF DATA FILE
;12 - SECOND WORK FILE
;6 - 14 RUN-TIME COBOL FILES
;
;THE NEXT COMMAND IDENTIFIES THIS TASK AS A COMPILE AND GO SYSTEM
;TO THE RESIDENT COMPILER CODE
;
GBLDEF = TASKSW:1
GBLDEF = MSGLUN:2
GBLDEF = WRKLUN:3
GBLDEF = ACCLUN:4
GBLDEF = DISLUN:5
GBLDEF = XLUN:7
GBLDEF = INLUN:10
GBLDEF = OUTLUN:0                ;NOT USED BUT NEEDS TO BEDEFINED
GBLDEF = MUL8UF:0                ;NOT USED BUT NEEDS TO BE DEFINED
GBLDEF = SRCLUN:6
GBLDEF = CPYLUN:7
GBLDEF = OBJLUN:10
GBLDEF = INTLUN:11
GBLDEF = LSTLUN:12
GBLDEF = CRFLUN:13
GBLDEF = MAPLUN:13
GBLDEF = WF2LUN:14       ;SECOND WORK FILE
GBLDEF = FIRLUN:6
GBLDEF = MAXFLS:16
GBLDEF = TTDEV:41113     ;KB FOR RSTS/E
/
```

GO TO RESERVE WORD PROBLEM (SPR 11-8169 BR & RB)

A GO TO statement whose destination is a reserved word causes compiler to loop.

The problem will be corrected in module AGOP. This will be considered for the next release of PDP-11 COBOL.

UNSTRING VERB USING NONINTEGER DATA ITEMS (SPR 11-8169 BR & RB)

An UNSTRING statement containing a noninteger numeric receiving field causes fatal error #222. No such diagnostic should be issued as this is a legitimate operand.

The compiler module UNST will be corrected to permit a noninteger numeric receiving field in an UNSTRING statement. This will be considered for the next release of PDP-11 COBOL.

COMPILER DOES NOT ACCEPT SOURCE INPUT FROM CARD READER (SPR 11-8595 BR)

COBOL compiler does not accept source input from card reader. Error "Open Failure on Output File" results from:

```
CBL>,KB:=CR:/CVF
```

Give source input file a file name in command string.

Example:

```
CBL>,KB:=CR:A/CVF
```

RECORD SIZE APPEARS AS FOUR ASCII BYTES (SPR 11-9108 BR)

1. Record size appears as four ASCII bytes in magtape files.

   ANSI magtape conventions prescribe four ASCII bytes to define record size. See PDP-11 COBOL User's Guide Version 2, DEC-11-LCUGA-B-D, section 6.2.5.

2. Magtape record blocking not expected if BLOCK CONTAINS 512 CHARACTERS appears in COBOL program.

   Default magtape blocking uses a minimum physical block size of 512 bytes. Each such block contains one or more complete (no record spanning) logical records. See PDP-11 COBOL User's Guide Version 2 DEC-11-LCUGA-B-D, section 6.2.5.

STACK OVERFLOW (SPR 11-8684A BR)

COBOL-11 terminates with stack overflow.

The solution is to build the compiler with following parameters given following values in file CBLRSX.CMD :

```
STACK=1Ø24
EXTSCT=WFBLK:12ØØØ
EXTSCT=WFEND:32ØØØ
```

APPLY MAP8 (SPR 11-9052 RH)

The COBOL compiler does not accept APPLY MAP8.. in the I/O CONTROL paragraph. It outputs the error messages SYNTAX ERROR (#34) and UNKNOWN WORD (#162). It does accept APPLY MAP4.., APPLY MAP5.. , APPLY MAP6.. , and APPLY MAP7.. correctly.

This problem will be considered for future release. In the meantime, APPLY MAP7.. should be used in place of APPLY MAP8..

COPY VERB PROBLEM IN "WORKING-STORAGE" SECTION (SPR 11-9208 FI)

If a COPY statement is the first nonblank line in the WORKING-STORAGE section, certain items in WORKING-STORAGE defined by the COPY library statements are not being allocated correctly.

The problem is caused by the COPY verb processor which is erroneously saving and restoring an internal data item needed for allocation of WORKING-STORAGE items found in the COPY library file.

This can be solved effectively by including at least one data item definition in WORKING-STORAGE before the COPY statement. A permanent fix for this problem is under consideration.

ERRONEOUS "PERFORM" ERROR MESSAGES (SPR 11-9880 FI)

The COBOL runtime system outputs incorrect error messages for certain errors in PERFORM processing.

1. The message, MOUNT NEXT INPUT REEL, is erroneously printed whenever a PERFORM is executed recursively.

2. The message, NOT OPEN, is erroneously printed whenever an exit from a PERFORMed procedure is made and an included PERFORMed procedure is still active.

These problems will be addressed in any future releases of COBOL-11.

COMP ITEM SYNCHRONIZATION (SPR 11-10120 BR)

Group level moves may produce unexpected results.

Word alignment is enforced during storage allocation of computational items, but not enforced during storage allocation of group items. A group move is treated as an alphanumeric move. If the groups contain computational items, there is no enforcement of synchronization of the computational items.

EXAMPLE: GROUP-1 takes up one word. FILLER takes up one byte. GROUP-2 begins on a byte boundary, but since it is computational, ITEM-2 is word aligned. Therefore, the storage addresses of GROUP-2 and ITEM-2 differ by one byte; and the sizes of GROUP-2 and ITEM-2 differ. This can be verified by compiling the example program with the /MAP switch. This one byte discrepancy in storage allocation and size produces the unexpected results.

```
COBOL 02.00 SRC:BASIS.CBL;4                          13:55:46 PAGE 001

  00001         IDENTIFICATION DIVISION.
  00002         PROGRAM-ID.     DEMO.
  00003         ENVIRONMENT DIVISION.
  00004         DATA DIVISION.
  00005         WORKING-STORAGE SECTION.
  00006         01 DEMO-1.
  00007             03 GROUP-1.
  00008                 05 ITEM-1      PIC 999 COMP VALUE 123.
  00009             03 FILLER          PIC 9.
  00010             03 GROUP-2.
  00011                 05 ITEM-2      PIC 999 COMP VALUE ZERO.
  00012         01 DISPLAY-ITEM        PIC ZZZ999.
  00013         PROCEDURE DIVISION.
  00014         1.
  00015             PERFORM DISPLAY-ITEMS.
  00016             MOVE GROUP-1 TO GROUP-2.
  00017             PERFORM DISPLAY-ITEMS.
  00018             STOP RUN.
  00019         DISPLAY-ITEMS.
  00020             MOVE ITEM-1 TO DISPLAY-ITEM.
  00021             DISPLAY "ITEM-1: "  DISPLAY-ITEM.
  00022             MOVE ITEM-2 TO DISPLAY-ITEM.
  00023             DISPLAY "ITEM-2: "  DISPLAY-ITEM.
```

```
NO ERRORS
CBL -- LOADING
ITEM-1:     123
ITEM-2:     000
ITEM-1:     123
ITEM-2:     192
CBL -- I000027 STOP RUN
MCR>
```

LIMIT IN PERFORM ITERATION COUNTS (SPR 11-10121 BR)

PERFORM iteration count, greater than 32767, does not work.

The maximum value of a PERFORM iteration count is 32767. This is a permanent restriction. This restriction will be described in an updated COBOL-11 User's Guide.

TERMINAL FILENAME (SPR 11-10343 RB)

BATCH will not work when inputting from KB: without use of filename.

When compiling source input from device KB:, specify a filename, e.g.:

```
CBL>x,y=KB:Z
```

where Z is the filename in question.

This will be considered in any future release of COBOL-11.

If the COBOL source deck is part of the BATCH stream, the BATCH processor builds the COBOL command string and does not include a dummy file name after KB:.

To do a compilation in BATCH with the source deck in the stream, first use PIP to copy the source deck to a disk file before the $COBOL command.

EFFICIENCY TECHNIQUES

During the long life of the COBOL-11 product, several techniques have been discovered that tend to make for more efficient COBOL-11 programs.

1. Always configure COBOL at 28KW, as this lessens paging.

2. Use COMPUTATIONAL rather than DISPLAY usage to describe numeric data in files, as this results in approximately a 50% reduction in the space required to store numeric data on disk.

3. Use COMPUTATIONAL usage data in numeric operations. Savings of up to 90% in execution speed may result.

4. Match operand usage in IF statements, to save the time required to translate the operands to the same usage.

5. If possible, match scaling (number of decimal places) of numeric operands, to save the time required to do this.

6. Use indexing rather than subscripting. Indices contain the actual table offset, rather than having to compute the offset from the occurrence number, as is the case with subscripts. Even relative indexing is faster than subscripting.

7. When using RELATIVE files, write the last record first, to save the time required to do multiple file extends.

*8. Specify file clustersize appropriately. For example, VALUE OF ID "REPORT.DAT/CL:256" is appropriate for a large data file. Negative clustersize values are also supported.

*9. In SEQUENTIAL files containing COMPUTATIONAL data, position at least one COMP item at the beginning of the record to outwit the RTSLIB scan for such items.

10. Use the SYM:4 switch to speed up compilations.

11. Maintain a tight working set. Transferring control to widely separated parts of the program will increase paging.

12. Restrict the usage of the COPY verb. Inclusion of large amounts of unnecessary code makes for larger programs, increasing paging overhead.

*RSTS/E only.

*13. Restrict the use of interactive COBOL programs on small systems. A good configuration guideline is to count every COBOL job as three BASIC-PLUS jobs; i.e., 16KW of physical user space for every COBOL job.

14. Asking for maps and cross reference listings slows a COBOL compilation appreciably. Do not ask unless they are needed.

15. Do not put all your files on the system disk. Spreading files across several disks allows for overlapped seeks and therefore faster file processing.

*RSTS/E only.

COBOL-11 V2 for IAS V1.1, RSTS/E V06A-02, V06B-02
and RSX-11D V6B

RELATIVE I/O (SPR 11-10324 FI)

When a write or rewrite causes an extend of the BITMAP for a relative file while a file extend is needed to contain the data records, the OTS forgets to update BITMAP for the new record. No error diagnosis is given until a retrieval of the record is attempted, whereupon a NON-EXISTENT RECORD error is returned. This patch will fix the problem.

1. ASSEMBLE THE ATTACHED PROGRAM AND NAME THE
   OBJECT FILE OTSPAT.OBJ.

2. MODIFY THE .ODL FILE FOR THE COBOL ON YOUR SYSTEM
   ON IAS OR RSX11D THE FILE IS CBLBLD.ODL
   ON RSTS/E THE FILE IS CBRSTS.ODL

   MODIFY THE .FCTR LINE IN THE ODL FILE THAT LOOKS LIKE

   .FCTR ...EXEC-OTS...

   ADD A REFERENCE TO OTSPAT SO THAT THE LINE LOOKS LIKE

   .FCTR ...EXEC-OTS-OTSPAT...

   IT IS VERY IMPORTANT THAT THE STRING "OTS-OTSPAT"
   APPEAR AS SHOWN.

3. RE-TASK BUILD THE COBOL TASK USING THE MODIFIED ODL FILE

4. RE-INSTALL THE NEW TASK IMAGE ON YOUR SYSTEM.

```
        .TITLE OTSPAT,<PATCH FOR OTS RELATIVE I/O BUG>
        ;
        ;THIS PATCH FIXES BUGS IN RELATIVE FILE SUPPORT
        ;WHEN BIT MAP RECORDS DO NOT GET UPDATED CORRECTLY DURING
        ;RELATIVE RANDOM REWRITE,WRITE ETC., AND WHEN A SIMULTANEOUS BITMAP
        ;EXTEND MUST BE DONE.
        ;THE PATCH CONSISTS OF AN ADDITIONAL CALL TO .RPRTC AFTER THE CALL TO
        ;.RPUT IS MADE
        ;
                .PSECT  OTS,GBL,I,RO,REL,OVR
                .GLOBL  .RPRTC,R.BTVL
                STATOK=OTS+426

        OTS:
                .=OTS+5122
                JSR     PC,WRRPAT       ;CALL THE PATCH
                .=OTS+6666
        WRRPAT:
                MOVB    #1,R.BTVL(R0)
                JSR     PC,.RPRTC
                JSR     PC,STATOK
                RTS     PC
                .END
```

COBOL-11 INSTALLATION PROCEDURE

This article describes the procedure for installing COBOL-11 Version 2 on RSTS/E V06B-02 from the COBOL distribution kit. If you have previously installed the COBOL compiler and SORT11 program under V06A-02, it is only necessary to ensure that you have the V06B-02 version of the RTSLIB Run-Time System installed. You do not have to re-install COBOL and SORT11 unless you are using a new system disk for V06B-02.

Before building COBOL, ensure that the following requirements are satisfied:

1. The Run-Time System RTSLIB must be on the system and added as an auxiliary Run-Time System (see Article 20.1.1).

2. The default swap maximum must be at least as large as the size of the COBOL compiler (specified during the build procedure below).

3. The size of the COBOL compiler must not be greater than the available contiguous user space on the system.

4. The total of the COBOL compiler size and the RTSLIB size (fixed at 4K words) must not be greater than the available user space on the system.

To build COBOL, the LNGBLD program is run. The LNGBLD program creates an account on the system disk. Upon completion of the build procedure, the account is zeroed and deleted.

The LNGBLD procedures allow the size and name of the COBOL compiler to be altered. This facility enables systems to have both a large and small COBOL compiler at the same time. (LNGBLD creates only one compiler at a time.)

The following procedure describes how COBOL and SORT11 are built using a magtape distribution medium. The procedure for building from a DECtape or disk cartridge medium is similar. The order numbers for the respective media are listed below:

Medium	Order Number
Magtape (7-track)	DEC-11-LCBSA-C-MB7
Magtape (9-track)	DEC-11-LCBSA-C-MB9
DECtape	DEC-11-LCBSA-C-UB1, -UB2
Disk cartridge	DEC-11-LCBSA-C-HB

COBOL-11 V2
COBOL-11 INSTALLATION PROCEDURE

NOTE

The following procedure replaces any file named BIGTKB.TSK in account [1,2] by a file of the same name from the distribution medium. In order to preserve [1,2]BIGTKB.TSK, rename it to a different name before starting the following procedure.

If the distribution medium is magtape, ensure that there is no write enable ring on the reel. Mount the distribution medium on an available drive and write lock the drive (if DECtape) or write protect it (if disk cartridge). Ready the drive by positioning the magtape at load point and setting the drive to ON-LINE, or by setting the DECtape drive to REMOTE, or by setting the disk drive to the RUN condition.

Log into account [1,2] and run the LNGBLD program as shown in the following example. All user supplied input is underlined and should be terminated with carriage return.

```
RUN LNGBLD
LNGBLD  V06B-02 RSTS V06B-02 Timesharing
WHAT LANGUAGE IS TO BE BUILT? COBOL
FROM WHAT DEVICE(S) WILL THE FILES COME? MT0:
ACCOUNT (1,7) WILL BE CREATED AND USED FOR THE BUILD
THIS ACCOUNT WILL BE DELETED AFTER THE BUILD
↑C
HELLO1/7;A


Ready

ASSIGN MT0:.DOS

RUN $PIP
PIP     V06B-03 RSTS V06B-02 Timesharing
#SY:=MT0:[128,128]/CO
#↑Z

Ready

DEASSIGN

Ready
```

COBOL-11 V2
COBOL-11 INSTALLATION PROCEDURE


```
!********** BUILD THE COBOL LANGUAGE PROCESSOR

RUN CBLBLD
DEFAULTS FOR THE COBOL BUILD ARE:
        NAME            COBOL
        SIZE            23K WORDS
        SAVE MAP?       NO

ANY CHANGES?
```

At this point in the procedure, either YES or NO must be typed. If NO is typed, the program builds COBOL with the defaults listed in the printout. If YES is typed, the program prints a series of three questions which allow the defaults to be changed.

The first question concerns the size of the COBOL job image. Typing NO establishes the default size and causes the second question to be printed. Typing YES causes the program to print an additional question concerning the size. The size can be between 23K and 28K words provided that the available user job space is at least equal to the sum of the COBOL size and the size of RTSLIB. If enough user job space is not available and a user attempts to run COBOL, RTSLIB prints the PROGRAM TOO BIG error message. A larger size for COBOL increases execution efficiency because the number of overlays is reduced. A smaller size requires less memory and swapping storage.

The second question allows the name of the COBOL compiler to be changed. Typing NO to the question establishes the default name. Typing YES causes the program to print an additional question in response to which a name between one and six characters can be typed. The name specified is used for the COBOL module and for the program which loads the COBOL module.

The third question concerns saving the load map file. Typing NO results in the map being deleted. Typing YES saves the map under account [1,2] in a file called nnnnnn.MAP where nnnnnn is the one to six characters assigned to the COBOL module name. (If YES is typed, a message giving the file name and account is printed on the next line for documentation purposes.)

The dialogue continues as shown in the following sample printout.

```
ANY CHANGES? YES
        CHANGE SIZE? YES
                CHANGE IT TO? 28
        CHANGE NAME? NO
        SAVE MAP? YES
        THE MAP WILL BE SAVED AS COBOL.MAP ON ACCOUNT [1,2].

Ready
```

COBOL-11 V2
COBOL-11 INSTALLATION PROCEDURE


```
RUN $PIP
PIP     V06B-03 RSTS V06B-02 Timesharing
#$BIGTKB.TSK=BIGTKB.TSK/BL
#$BIGTKB.TSK<232>/RE
#↑Z

Ready

RUN$UTILTY
UTILTY  V06B-03 RSTS V06B-02 Timesharing
#NAME RTSLIB=$BIGTKB.TSK
#LOAD RTSLIB
#↑Z

Ready

RUN$BIGTKB.TSK

TKB>@CBRSTS
TKB -- *DIAG*-21 PHS2   HT

TKB -- *DIAG*-22 EXEC

TKB -- *DIAG*-22 RRFCS

TKB>@SRTRST
TKB -- *DIAG*-21 SORTP  FA.ENB

TKB -- *DIAG*-21 SORTP  FA.POS

TKB -- *DIAG*-21 SORTP  FA.RWD

TKB -- *DIAG*-21 SORTM  FA.ENB

TKB -- *DIAG*-21 SORTM  FA.POS

TKB -- *DIAG*-21 SORTM  FA.RWD

TKB -- *DIAG*-22 SORTS1

TKB>↑Z

Ready
```

COBOL-11 V2
COBOL-11 INSTALLATION PROCEDURE


```
OLD COBOL

Ready

10      DATA    COBOL,$COBOL.TSK,RTSLIB,-1
COMPILE SY0:$COBOL

Ready

OLD SORT11

Ready

COMPILE SY0:$SORT11

Ready

NAME "$COBOL.BAC" AS "COBOL.BAC<232>"

Ready

NAME "$COBOL.TSK" AS "COBOL.TSK<232>"

Ready

NAME      "$SORT11.TSK" AS "$SORT11.TSK<232>"

Ready

NAME      "$SORT11.BAC" AS "$SORT11.BAC<232>"

Ready
```

```
!****RUN A COBOL TEST PROGRAM *****
!
RUN $COBOL

CBL>XTST01=XTST01
CBL -- 4    ERRORS
CBL -- LOADING
COMPARE THE FOLLOWING TWO ROWS:
A    ABCD A
A    ABCD A

COMPARE THE FOLLOWING TWO ROWS:
AAE1212221B1B1K01201202201B01B01K
AAE1212221B1B1K01201202201B01B01K


A POPULAR   POEM

MARY
HAD
A
LITTLE
LAMB,
ITS
FLEECE
WAS
WHITE
AS
SNOW,
AND
EVERYWHERE
THAT
MARY
WENT,
THE
LAMB
WAS
SURE
TO
GO.
IT
FOLLOWED
HER
TO
SCHOOL
ONE
DAY,
WHICH
WAS
AGAINST
THE
RULE.
```

COBOL-11 V2
COBOL-11 INSTALLATION PROCEDURE


```
IT
MADE
THE
CHILDREN
LAUGH
AND
END
CBL -- I000027 STOP RUN

CBL>↑Z

Ready

!!!!!!!!!!
!!!!!!!!!!
!!!!!!!!!!
!!!!!!!!!!
!!!!!!!!!!

RUN $PIP
PIP     V06B-03 RSTS V06B-02 Timesharing
#SY:$<REFRMT.OBJ,COBRG.OBJ/CO
#↑Z

Ready

BUILD COMPLETE - ZEROING ACCOUNT (1,7) - DELETING ACCOUNT (1,7)
Ready
```

RPG-II V7.06 for RSTS/E V06B-02
RPG-II Re-released for RSTS/E V06B-02

RPG-II RE-RELEASED FOR RSTS/E V06B-02

RPG-II has been re-released for RSTS/E V06B-02. The new version of RPG-II is V7.06 (not V7.04). If RPG has been previously installed on your RSTS/E system under V06A-02, you must obtain a new RPG distribution kit and re-install RPG.

DOCUMENTATION UPDATE (SPR 11-8257 FI)

PROBLEM: The SORT-11 Reference Manual does not correctly describe the output files of SORTA or SORTI sort processes when SORT-11 is running on RSTS/E.

DISPOSITION: See documentation update on the following page.

DOCUMENTATION UPDATE

SORT-11 Reference Manual DEC-11-USTMA-B-D

1.3.6 RSTS/E SORT-11 Record Formats

The format of records written by SORT-11 on RSTS/E is somewhat different than that produced by SORT-11 on RSX and IAS. While the actual data remains the same, SORT-11 on RSTS/E may write what are known as DOS formatted binary records (Format 3) if binary data is to be included in the file.

There are three possible record formats in RSTS/E COBOL storage files:

Format 1: This is the standard Formatted ASCII record, consisting of data bytes, beginning on an odd or even address, followed by a carriage return (octal 015) byte, followed by a line feed (octal 012) byte.

Format 2: Format 2 records are similar to Format 1 records, except that three null (octal 000) bytes are prefixed to the record.

Format 3: Format 3 records are DOS Formatted Binary records. The first byte is an octal 001, the second byte is null (000), the third byte is the low-order byte of the byte count, and the fourth byte is the high-order byte of the byte count. The data bytes follow. The last byte is a checksum byte, calculated on output and verified on input. The byte count is the number (in binary) of data bytes plus 4 (the number of header bytes). It does not include the checksum. The checksum is calculated so that the (8-bit) sum of all bytes, including header and checksum, is zero.

For variable-length record files, RTSLIB (FCS emulator) will scan the data bytes. If no bytes fall in the ranges of octal 000 to octal 037, inclusive, or octal 200 to octal 377, inclusive, a Format 1 record will be written; otherwise, a Format 3 record will be written.

For fixed length record files, a similar scan is employed, except that Format 2 records will be written for data with all bytes in the range of octal 040 to octal 177, inclusive. Otherwise, a Format 3 record will be written.

7.2 Commonly Used System Programs (CUSPs) Status Report

This report reflects the current Version/Edit level of all of the Commonly Used System Programs (CUSPs) supplied as part of the standard RSTS/E V06B-02 System Library Programs release.

System Utilities Package

BUILD	V06B-03	DIRECT	V06B-03	DISPLY	V06B-03	VT05	V06B-03
VT50	V06B-03	DSKINT	V06B-03	GRIPE	V06B-03	INIT	V06B-03
INUSE	V06B-03	LOGIN	V06B-03	LOGOUT	V06B-03	MONEY	V06B-03
ODT	V06B-03	PLEASE	V06B-03	PRIOR	V06B-03	QUOLST	V06B-03
REACT	V06B-03	REORDR	V06B-03	SHUTUP	V06B-03	SYSCAT	V06B-03
SYSTAT	V06B-03	TALK	V06B-03	TTYSET	V06B-03	UMOUNT	V06B-03
UTILTY	V06B-03	UTILT1	V06B-03				

Data Manipulation Package

COPY	V06B-03	EDIT	V06B-03	EDITCH	V06B-03	FILCOM	V06B-03
FLINT	V06B-03	PIP	V06B-03	PIPEXT	V06B-03	RUNOFF	V06B-03

Error Control Package

ANALY1	V06B-03	ANALYS	V06B-03	ERRCPY	V06B-03	ERRDET	V06B-03
ERRDIS	V06B-03	ERRINT	V06B-03				

Backup Package

BACCOM	V06B-03	BACCON	V06B-03	BACDEL	V06B-03	BACDIR	V06B-03
BACDMP	V06B-03	BACDSK	V06B-03	BACENT	V06B-03	BACFRM	V06B-03
BACLAB	V06B-03	BACLOD	V06B-03	BACLST	V06B-03	BACMNT	V06B-03
BACPRM	V06B-03	BACKTO	V06B-03	BACKUP	V06B-03		

Spooling and Operator Services Package

OPSER	V06B-03	OPSRUN	V06B-03				
QUE	V06B-03	QUEMAN	V06B-03	QUMRUN	V06B-03		
CHARS	V06B-03	SPOOL	V06B-03	SPLIDL	V06B-03	SPLRUN	V06B-03
BATDCD	V06B-03	BATCH	V06B-03	BATIDL	V06B-03	BATDEC	V06B-03
BATRUN	V06B-03						

8.0 Software Product Descriptions (SPDs) and Option Bulletins

8.1 Software Product Descriptions (SPD's)

Periodically, new or revised Software Product Description (SPD) bulletins will appear in the Software Dispatch for various software products marketed by DIGITAL. This section of the notebook is provided so that each installation may have a convenient place to file these SPDs for future reference.

Every SPD has a unique number, a sample of which is shown below:

SPD 14.52.0

This number appears on the first page of the SPD bulletin, at the far right end of the PRODUCT NAME: line.

Note that the last part of the SPD number is the revision level, with 0 being assigned for original issues of SPDs.

8.2 Option Bulletins

Periodically, new or revised Option Bulletins will appear in the Software Dispatch for various products marketed by DIGITAL. This section of the release notes is provided so that each installation may have a convenient place to file these bulletins. There is no defined filing scheme for Option Bulletins.

Appendix A SYSGEN Editor Manual

RSTS/E system generations are normally performed under a RSTS/E SYSGEN monitor configured for all supported disk and tape devices. During the system generation process the RT11 emulator serves as the keyboard monitor and Run-Time System for the various utility programs. In some special cases, it may be necessary to use a text editor during SYSGEN to modify batch or configuration files. Since the BASIC-PLUS Run-Time System is normally not available at this time, the standard RSTS/E EDIT program cannot be used. To allow modification of files during system generation, the RT11 Editor (which runs under the emulator) is included in the RSTS/E distribution kits. RT11 EDIT is supplied only for SYSGEN purposes.

The RT11 EDIT manual is reproduced on the following pages. This information was extracted verbatim from the RT11 System Reference Manual and certain sections do not apply in the RSTS/E environment. Specifically, the display editor features presented in Section 3.7 do not apply and should be ignored.

CHAPTER 3

TEXT EDITOR

The Text Editor (EDIT) is used to create and modify ASCII source files so that these files can be used as input to other system programs such as the assembler or BASIC. Controlled by user commands from the keyboard, EDIT reads ASCII files from a storage device, makes specified changes and writes ASCII files to a storage device or lists them on the line printer or console terminal. EDIT allows efficient use of VT-11 display hardware, if this is part of the system configuration.

The Editor considers a file to be divided into logical units called pages. A page of text is generally 50-60 lines long (delimited by form feed characters) and corresponds approximately to a physical page of a program listing. The Editor reads one page of text at a time from the input file into its internal buffers where the page becomes available for editing. Editing commands are then used to:

Locate text to be changed,

Execute and verify the changes,

Output a page of text to the output file,

List an edited page on the line printer or console terminal.

3.1 CALLING AND USING EDIT

To call EDIT from the system device type:

```
R EDIT
```

and the RETURN key in response to the dot (.) printed by the monitor. EDIT responds with an asterisk (*) indicating it is in command mode and awaiting a user command string.

Type CTRL C to halt the Editor at any time and return control to the monitor. To restart the Editor type .R EDIT or the .REENTER command in response to the monitor's dot. The contents of the buffers are lost when the Editor is restarted.

3.2 MODES OF OPERATION

Under normal usage, the Editor operates in one of two different modes: Command Mode or Text Mode. In Command Mode all input typed on the keyboard is interpreted as commands instructing the Editor to perform some operation. In Text Mode all typed input is interpreted as text to replace, be inserted into, or be appended to the contents of the Text Buffer.

Immediately after being loaded into memory and started, the Editor is in Command Mode. An asterisk is printed at the left margin of the console terminal page indicating that the Editor is waiting for the user to type a command. All commands are terminated by pressing the ALTMODE key twice in succession. Execution of commands proceeds from left to right. Should an error be encountered during execution of a command string, the Editor prints an error message followed by an asterisk at the beginning of a new line indicating that it is still in Command Mode and awaiting a legal command. The command in error (and any succeeding commands) is not executed and must be corrected and retyped.

Text mode is entered whenever the user types a command which must be followed by a text string. These commands insert, replace, exchange, or otherwise manipulate text; after such a command has been typed, all succeeding characters are considered part of the text string until an ALTMODE is typed. The ALTMODE terminates the text string and causes the Editor to reenter Command Mode, at which point all characters are considered commands again.

A special editing mode, called Immediate Mode, can be used whenever the VT-11 display hardware is running. This mode is described in Section 3.7.2.

3.3 SPECIAL KEY COMMANDS

The EDIT key commands are listed in Table 3-1. Control commands are typed by holding down the CTRL key while typing the appropriate character.

Table 3-1
EDIT Key Commands

Key	Explanation
ALTMODE	Echoes $. A single ALTMODE terminates a text string. A double ALTMODE executes the command string. For example, *GMOV A,B$-1D$$
CTRL C	Echoes at the terminal as ↑C and a carriage return. Terminates execution of EDIT commands, and returns to monitor Command Mode. A double CTRL C is necessary when I/O is in progress. The REENTER command may be used to restart the Editor, but the contents of the text buffers are lost.

(continued on next page)

Table 3-1 (cont.)
EDIT Key Commands

Key	Explanation
CTRL O	Echoes ↑O and a carriage return. Inhibits printing on the terminal until completion of the current command string. Typing a second CTRL O resumes output.
CTRL U	Echoes ↑U and a carriage return. Deletes all the characters on the current terminal input line. (Equivalent to typing RUBOUT back to the beginning of the line.)
RUBOUT	Deletes character from the current line; echoes a backslash followed by the character deleted. Each succeeding RUBOUT typed by the user deletes and echoes another character. An enclosing backslash is printed when a key other than RUBOUT is typed. This erasure is done right to left up to the last carriage return/line feed combination. RUBOUT may be used in both Command and Text Modes.
TAB	Spaces to the next tab stop. Tab stops are positioned every eight spaces on the terminal; typing the TAB key causes the carriage to advance to the next tab position.
CTRL X	Echoes ↑X and a carriage return. CTRL X causes the Editor to ignore the entire command string currently being entered. The Editor prints a <CR><LF> and an asterisk to indicate that the user may enter another command. For example: *IABCD EFGH^X * A CTRL U would only cause deletion of EFGH; CTRL X erases the entire command.

3.4 COMMAND STRUCTURE

EDIT commands fall into six general categories:

Category	Commands	Section
Input/Output	Edit Backup	3.6.1.3
	Edit Read	3.6.1.1
	Edit Write	3.6.1.2
	End File	3.6.1.9
	Exit	3.6.1.10
	List	3.6.1.7
	Next	3.6.1.6
	Read	3.6.1.4
	Verify	3.6.1.8
	Write	3.6.1.5
Pointer location	Advance	3.6.2.3
	Beginning	3.6.2.1
	Jump	3.6.2.2

Search	Find	3.6.3.2
	Get	3.6.3.1
	Position	3.6.3.3
Text modification	Change	3.6.4.4
	Delete	3.6.4.2
	eXchange	3.6.4.5
	Insert	3.6.4.1
	Kill	3.6.4.3
Utility	Edit Console	3.7.1
	Edit Display	3.7.1
	Edit Lower	3.6.5.6
	Edit Upper	3.6.5.6
	Edit Version	3.6.5.5
	Execute Macro	3.6.5.4
	Macro	3.6.5.3
	Save	3.6.5.1
	Unsave	3.6.5.2
Immediate Mode	ALTMODE	3.7.2
	CTRL D	3.7.2
	CTRL G	3.7.2
	CTRL N	3.7.2
	CTRL V	3.7.2
	RUBOUT	3.7.2

The general format for the first five categories of EDIT commands is:

```
nCtext$
```

or

```
nC$
```

where n represents one of the legal arguments listed in Table 3-2, C is a one- or two-letter command, and text is a string of successive ASCII characters.

As a rule, commands are separated from one another by a single ALTMODE; however, if the command requires no text, the separating ALTMODE is not necessary. Commands are terminated by a single ALTMODE; typing a second ALTMODE begins execution. (ALTMODE is used differently when Immediate Mode is in effect; Section 3.7.2 details its use in this case.)

The format of Display Editor commands is somewhat different from the normal editing command format, and is described in Section 3.7.

3.4.1 Arguments

An argument is positioned before a command letter and is used either to specify the particular portion of text to be affected by the command or to indicate the number of times the command should be performed. With some commands, this specification is implicit and no arguments are needed; other editing commands require an argument. Table 3-2 lists the formats of arguments which are used by commands of this second type.

Table 3-2
Command Arguments

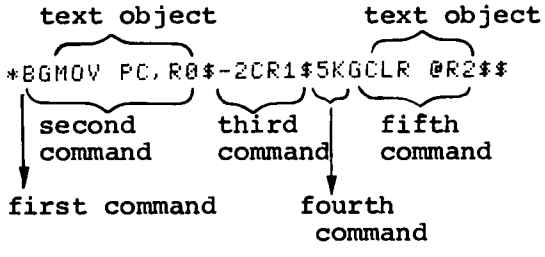
Format	Meaning
n	n stands for any integer in the range -16383 to +16383 and may, except where noted, be preceded by a + or -. If no sign precedes n, it is assumed to be a positive number. Whenever an argument is acceptable in a command, its absence implies an argument of 1 (or -1 if only the - is present).
0	0 refers to the beginning of the current line.
/	/ refers to the end of text in the current Text Buffer.
=	= is used with the J, D and C commands only and represents -n, where n is equal to the length of the last text argument used.

The roles of all arguments are explained more specifically in following sections.

3.4.2 Command Strings

All EDIT command strings are terminated by two successive ALTMODE characters. Spaces, carriage returns and line feeds within a command string may be used freely to increase command readability but are ignored unless they appear in a text string. Commands used to insert text can contain text strings that are several lines long. Each line is terminated with a <CR><LF> and the entire command is terminated with a double ALTMODE.

Several commands can be strung together and executed in sequence. For example,



Execution of a command string begins when the double ALTMODE is typed and proceeds from left to right. Except when they are part of a text string, spaces, carriage return, line feed, and single ALTMODES are ignored. For example:

```
*BGMOV R0$=CCLR R1$AV$$
```

may be typed as:

```
*B$ GMOV R0$
=CCLR R1$
A$ V$$
```

with equivalent execution.

3.4.3 The Current Location Pointer

Most EDIT commands function with respect to a movable reference pointer which is normally located between the most recent character operated upon and the next character in the buffer. At any given time during the editing procedure, this pointer can be thought of as representing the current position of the Editor in the text. Most commands use this pointer as an implied argument. Commands are available for moving the pointer anywhere in the text, thereby redefining the current location and allowing greater facility in the use of other commands.

3.4.4 Character- and Line-Oriented Command Properties

Edit commands are line-oriented or character-oriented depending on the arguments they accept. Line-oriented commands operate on entire lines of text. Character-oriented commands operate on individual characters independent of what or where they are.

When using character-oriented commands, a numeric argument specifies the number of characters that are involved in the operation. Positive arguments represent the number of characters in a forward direction (in relation to the pointer), negative arguments the number of characters in a backward direction. Carriage return and line feed characters are treated the same as any other character. For example, assume the pointer is positioned as indicated in the following text (↑ represents the current position of the pointer):

```
MOV     #VECT,R2<CR><LF>↑
CLR     @R2<CR><LF>
```

The EDIT command -2J backs the pointer by two characters.

```
MOV     #VECT,R2↑<CR><LF>
CLR     @R2<CR><LF>
```

The command 10J advances the pointer forward by ten characters and places it between the CR and LF characters at the end of the second line.

```
MOV     #VECT,R2<CR><LF>
CLR     @R2<CR>↑<LF>
```

Finally, to place the pointer after the "C" in the first line, a -14J command is used. The J (Jump) command is explained in Section 3.6.2.2.

```
MOV     #VECT↑,R2<CR><LF>
CLR     @R2<CR><LF>
```

When using line-oriented commands, a numeric argument represents the number of lines involved in the operation. The Editor recognizes a line of text as a unit when it detects a <CR><LF> combination in the text. When the user types a carriage return, the Editor automatically inserts a line feed. Positive arguments represent the number of lines forward (in relation to the pointer); this is accomplished by counting carriage return/line feed combinations beginning at the pointer. So, if the pointer is at the beginning of a line, a line-oriented command argument of +1 represents the entire line between the current pointer and the terminating line feed. If the current pointer is in the middle of the line, an argument of +1 represents only the portion of the line between the pointer and the terminating line feed.

For example, assume a buffer of:

```
MOV    ↑PC,R1<CR><LF>
ADD     #DRIV-.,R1<CR><LF>
MOV     #VECT,R2<CR><LF>
CLR     @R2<CR><LF>
```

The command to advance the pointer one line (1A) causes the following change:

```
MOV     PC,R1<CR><LF>
↑ADD    #DRIV-.,R1<CR><LF>
MOV     #VECT,R2<CR><LF>
CLR     @R2<CR><LF>
```

The command 2A moves the pointer over 2 <CR><LF> combinations:

```
MOV     PC,R1<CR><LF>
ADD     #DRIV-.,R1<CR><LF>
MOV     #VECT,R2<CR><LF>
↑CLR    @R2<CR><LF>
```

Negative line arguments reference lines in a backward direction (in relation to the pointer). Consequently, if the pointer is at the beginning of the line, a line argument of -1 means "the previous line" (moving backward past the first <CR><LF> and up to but not including the second <CR><LF>; if the printer is in the middle of a line, an argument of -1 means the preceding 1 1/2 lines. Assume the buffer contains:

```
MOV     PC,R1<CR><LF>
ADD     #DRIV-.,R1<CR><LF>
MOV     #VECT,R2<CR><LF>
CLR     @R2<CR><LF>
```

A command of -1A backs the pointer by 1 1/2 lines.

```
MOV     PC,R1<CR><LF>
↑ADD    #DRIV-.,R1<CR><LF>
MOV     #VECT,R2<CR><LF>
CLR     @R2<CR><LF>
```

Now a command of -1A backs it by only 1 line.

```
↑MOV    PC,R1<CR><LF>
 ADD    #DRIV-.,R1<CR><LF>
 MOV    #VECT,R2<CR><LF>
 CLR    @R2<CR><LF>
```

3.4.5 Command Repetition

Portions of a command string may be executed more than once by enclosing the desired portion in angle brackets (<>) and preceding the left angle bracket with the number of iterations desired. The structure is:

```
C1$C2$n<C3$C4$>C5$$
```

where C1, C2,...C5 represent commands and n represents an iteration argument. Commands C1 and C2 are each executed once, then commands C3 and C4 are executed n times. Finally command C5 is executed once and the command line is finished. The iteration argument (n) must be a positive number (1 to 16,383), and if not specified is assumed to be 1. If the number is negative or too large, an error message is printed. Iteration brackets may be nested up to 20 levels. Command lines are checked to make certain the brackets are correctly used and match prior to execution.

Essentially, enclosing a portion of a command string in iteration brackets and preceding it with an iteration argument (n) is equivalent to typing that portion of the string n times. For example:

```
*BGAAA$3<-DIB$-J>V$$
```

is equivalent to typing:

```
*BGAAA$-DIB$-J-DIB$-J-DIB$-JV$$
```

and:

```
*B3<2<AD>V>$$
```

is equivalent to typing:

```
*BADADVADADVADADV$$
```

The following bracket structures are examples of legal usage:

```
<<><<<><>>>>
<<<>>><><>
```

The following bracket structures are examples of illegal combinations which will cause an error message since the brackets are not properly matched:

```
><><
<<<>>
```

During command repetition, execution proceeds from left to right until a right bracket is encountered. EDIT then returns to the last left

bracket encountered, decrements the iteration counter and executes the commands within the brackets. When the counter is decremented to 0, EDIT looks for the next iteration count to the left and repeats the same procedures. The overall effect is that EDIT works its way to the innermost brackets and then works its way back again. The most common use for iteration brackets is found in commands such as Unsave, that do not accept repeat counts. For example:

```
*3<U>$$
```

Assume a file called SAMP (stored on device DK) is to be read and the first four occurrences of the instruction MOV #200,R0 on each of the first five pages are to be changed to MOV #244,R4. The following command line is entered:

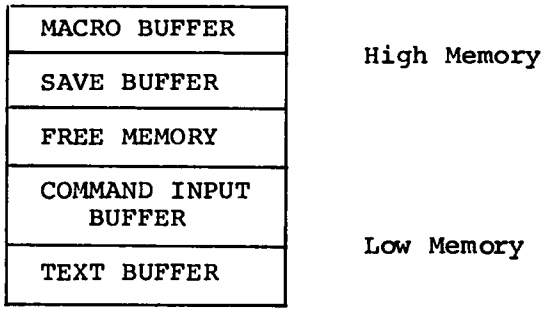
```
*EBSAMP$5<N4<BGMOV #200,R0$=J$3<G0$=C4$>>>EX$$
                          C
                    B
              A
```

The command line contains three sets of iteration loops (A,B,C) and is executed as follows:

Execution initially proceeds from left to right; the file SAMP is opened for input, and the first page is read into memory. The pointer is moved to the beginning of the buffer and a search is initiated for the character string MOV #200,R0. When the string is found, the pointer is positioned at the end of the string, but the =J command moves the pointer back so that it is positioned immediately preceding the string. At this point, execution has passed through each of the first two sets of iteration loops (A,B) once. The innermost loop (C) is next executed three times, changing the 0s to 4s. Control now moves back to pick up the second iteration of loop B, and again moves from left to right. When loop C has executed three times, control again moves back to loop B. When loop B has executed a total of 4 times, control moves back to the second iteration of loop A, and so forth until all iterations have been satisfied.

3.5 MEMORY USAGE

The memory area used by the Editor is divided into four logical buffers as follows:



The Text Buffer contains the current page of text being edited, and the Command Input Buffer holds the command currently being typed at the terminal. If a command currently being entered by the user is within 10 characters of exceeding the space available in the Command Buffer, the message:

```
* CB ALMOST FULL *
```

is printed. If the command can be completed within 10 characters, the user may finish entering the command; otherwise he should type the ALTMODE key twice to execute that portion of the command line already completed. The message is printed each time a character is entered in one of the last 10 spaces.

If the user attempts to enter more than 10 characters the message:

```
?CB FULL?
```

is printed and all commands typed within the last 10 characters are ignored. The user again has 10 characters of available space in which to correct the condition.

The Save Buffer contains text stored with the Save (S) command, and the Macro Buffer contains the command string macro entered with the Macro (M) command. (Both commands are explained in Section 3.6.5.)

The Macro and Save Buffers are not allocated space until an M or S command is executed. Once an M or S command is executed, a 0M or 0U (Unsave) command must be executed to return that space to the free area.

The size of each buffer automatically expands and contracts to accommodate the text being entered; if there is not enough space available to accommodate required expansion of any of the buffers, a "?*NO ROOM*?" error message is typed.

3.6 EDITING COMMANDS

3.6.1 Input/Output Commands

Input commands are used to create files and read them into the Text Buffer where they become available for editing or listing. Output commands cause text to be listed on the console terminal or line-printer or written out to a storage device. Some commands are specifically designed for either input or output functions, while a few commands serve both purposes.

Once editing is completed and the page currently in the Text Buffer is written to the output file, that page of text is unavailable for further editing until the file is closed and reopened.

3.6.1.1 Edit Read - The Edit Read command opens an existing file for input and prepares it for editing. Only one file can be open for input at a time.

The form of the command is:

ERdev:filnam.ext$

The string argument (dev:filnam.ext) is limited to 19 characters and specifies the file to be opened. If no device is specified, DK: is assumed. If a file is currently open for input, that file is closed; any edits made to the file are preserved.

Edit Read does not input a page of text nor does it affect the contents of the other user buffers (see Section 3.5.)

Edit Read can be used on a file which is already open to close that file for input and reposition EDIT at its beginning. The first Read command following any Edit Read command inputs the first page of the file.

Examples:

*ERDT1:SAMP.MAC$$	Opens SAMP.MAC on device DT1: for input.
*ERSOURCE$$	Opens SOURCE on device DK: for input.

3.6.1.2 Edit Write - The Edit Write command sets up a file for output of newly created or edited text. However, no text is output and the contents of the user buffers are not affected. Only one file can be open for output at a time. Any current output files are closed.

The form of the command is:

EWdev:filnam.ext[n]$

The string argument (dev:filnam.ext[n]) is limited to 19 characters and is the name to be assigned to the output file being opened. If dev: is not specified, DK: is assumed. [n] is optional and represents the length of the file to be opened. If not specified, one half the largest available space is used; if this is not adequate for the output file size, the EF and EX commands will not close the output file, and all edits will be lost. It is thus recommended that the [n] construction be used whenever there is doubt as to whether enough space is available on the device for the output file.

If a file with the same name already exists on the device, the old file is deleted when an EXit, End File or another Edit Write command is executed.

Examples:

*EWDK:TEST.MAC$$	Opens the file TEST.MAC on device DK: for output.
*EWFILE.BAS[11]$$	Opens the file FILE.BAS (allocating 11 blocks) on the device DK: for output.

3.6.1.3 Edit Backup - The Edit Backup command is used to open an existing file for editing and at the same time create a backup version of the file. Any currently open file will be closed. No text is read or written with this command.

The form of the command is:

```
EBdev:filnam.ext[n]$
```

The device designation, filename and extension are limited to 19 characters. If dev: is not specified, DK: is assumed. [n] is optional and represents the length of the file to be opened; if not specified, one-half the largest available space is used.

The file indicated in the command line must already exist on the device designated since text will be read from this file as input. At the same time, an output file is opened under the same filename and extension. After an EB command has been successfully executed, the original file (used as input) is renamed with the current filename and a .BAK extension; any previous file with this filename and a .BAK extension is deleted. The new output file is closed and assigned the name as specified in the EB command. This renaming of files takes place whenever an Exit, End File, Edit Read, Edit Write or Edit Backup command is executed.

Examples:

*EBSY:BAS1.MAC$$	Opens BAS1.MAC on SY. When editing is complete, the old BAS1.MAC becomes BAS1.BAK and the new file becomes BAS1.MAC. Any previous version of BAS1.BAK is deleted.
*EBBAS2.BAS[15]$$	Opens BAS2.BAS on DK (allocating 15 blocks). When editing is complete, the old BAS2.BAS is labeled BAS2.BAK and the new file becomes BAS2.BAS. Any previous version of BAS2.BAK is deleted.

In EB, ER and EW commands, leading spaces between the command and the filename are illegal (the filename is considered to be a text string). All dev:file.ext specifications for EB, ER and EW commands conform to the RT-11 conventions for file naming and are identical to filenames entered in command strings used with other system programs.

3.6.1.4 Read - The Read command (R) causes a page of text to be read from the input file (previously specified in an ER or EB command) and appended to the current contents, if any, of the Text Buffer.

The form of the command is:

```
R
```

No arguments are used with the R command and the pointer is not moved. Text is input until one of the following conditions is met:

1. A form feed character, signifying the end of the page, is encountered. At this point, the form feed will be the last character in the buffer; or

2. The Text Buffer is within 500 characters of being full. (When this condition occurs, Read inputs up to the next <CR><LF> combination, then returns to Command Mode. An asterisk is printed as though the Read were complete, but text will not have been fully input); or

3. An end-of-file condition is detected, (the *EOF* message is printed when all text in the file has been read into memory and no more input is available).

The maximum number of characters which can be brought into memory with an R command is approximately 6,000 for an 8K system. Each additional 4K of memory allows approximately 8,000 additional characters to be input. An error message is printed if the Read exceeds the memory available or if no input is available.

3.6.1.5 Write - The Write command (W) moves lines of text from the Text Buffer to the output file (as specified in the EW or EB command). The format of the command is:

nW Write all characters beginning at the pointer and ending at the nth <CR><LF> to the output file.

-nW Write all characters beginning on the -nth line and terminating at the pointer to the output file.

0W Write the text from the beginning of the current line to the pointer.

/W Write the text from the pointer to the end of the buffer.

The pointer is not moved and the contents of the buffer are not affected. If the buffer is empty when the Write is executed, no characters are output.

Examples:

*5W$$	Writes the next 5 lines of text starting at the pointer, to the current output file.
*-2W$$	Writes the previous 2 lines of text, ending at the pointer, to the current output file.
*B/W$$	Writes the entire Text Buffer to the current output file.

3.6.1.6 Next - The Next command acts as both an input and output command since it performs both functions. First it writes the current Text Buffer to the output file, then clears the buffer, and finally reads in the next page of the input file. The Next command can be repeated n times by specifying an argument before the command. The command format is:

```
nN
```

Next accepts only positive arguments (n) and leaves the pointer at the beginning of the buffer. If fewer than n pages are available in the input file, all available pages are input to the buffer, output to the current file, and deleted from the buffer; the pointer is left positioned at the beginning of an empty buffer, and an error message is printed. This command is equivalent to a combination of the Beginning, Write, Delete and Read commands (B/W/DR). Next can be used to space forward, in page increments, through the input file.

Example:

*2N$$	Writes the contents of the current Text Buffer to the output file. Read and write the next page of text. Clear the buffer and then read in another page.

3.6.1.7 List - The List command prints the specified number of lines on the console terminal. The format of the command is:

nL	Print all characters beginning at the pointer and ending with the nth <CR><LF>.
-nL	Print all characters beginning with the first character on the -nth line and terminating at the pointer.
0L	Print from the beginning of the current line up to the pointer.
/L	Print from the pointer to the end of the buffer.

The pointer is not moved after the command is executed.

Examples:

*-2L$$	Prints all characters starting at the second preceding line and ending at the pointer.
*4L$$	Prints all characters beginning at the pointer and terminating at the 4th <CR><LF>.

Assuming the pointer location is:

```
MOVB   5(R1),@R2
ADD↑   R1,(R2)+
```

The command:

```
*-1L$$
```

Prints the previous 1 1/2 lines up to the pointer:

```
MOVB  5(R1),@R2
ADD
```

3.6.1.8 Verify - The Verify command prints the current text line (the line containing the pointer) on the terminal. The position of the pointer within the line has no effect and the pointer does not move. The command format is:

```
V
```

No arguments are used. The V command is equivalent to a 0LL (List) command.

Example:

```
*V$$                    The command causes the current line of
ADD     R1,(R2)+        text to be printed.
```

3.6.1.9 End File - The End File command closes the current output file. This command does no input/output operations and does not move the pointer. The buffer contents are not affected. The output file is closed, containing only the text previously output.

The form of the command is:

```
EF
```

No arguments are used. Note that an implied EF command is included in EW and EB commands.

3.6.1.10 EXit - The EXit command is used to terminate editing, copy the text buffer and the remainder of the input file to the output file, close input and output files, and return control to the monitor. It performs consecutive Next commands until the end of the input file is reached, then closes both the input and output files.

The command format is:

```
EX
```

No arguments are used. Essentially, Exit is used to copy the remainder of the input file into the output file and return to the monitor. Exit is legal only when there is an output file open. If an output file is not open and it is desired to terminate the editing session, return to the monitor with CTRL C.

NOTE

An EF or EX command is necessary in order to make an output file permanent. If CTRL C is used to return to the monitor without a prior execution of an EF command, the current output file is not saved. (It can however, be made permanent using the monitor CLOSE command; see Section 2.7.2.5.)

DIGITGRAPH
P. O Box 424
Tucson, Arizona 85717

An example of the contrasting uses of the EF and EX commands follows. Assume an input file, SAMPLE, contains several pages of text. The user wishes to make the first and second pages of the file into separate files called SAM1 and SAM2, respectively; the remaining pages of text will then make up the file SAMPLE. This can be done using these commands:

```
*EWSAM1$$
*ERSAMPLE$$
*RNEF$$
*EWSAM2$$
*NEF$$
*EWSAMPLE$EX$$
```

The user might note that the EF commands are not necessary in this example since the EW command closes a currently open output file before opening another.

3.6.2 Pointer Relocation Commands

Pointer relocation commands allow the current location pointer to be moved within the Text Buffer.

3.6.2.1 Beginning - The Beginning command moves the current location pointer to the beginning of the Text Buffer.

The command format is:

```
B
```

There are no arguments.

For example, assume the buffer contains:

```
MOVB    5(R1),@R2
ADD     R1,(R2)+
CLR     @R2
MOVB    6(R1),@R2
```

The B command:

```
*B$$
```

moves the pointer to the beginning of the Text Buffer:

```
↑MOVB   5(R1),@R2
 ADD    R1,(R2)+
 CLR    @R2
 MOVB   6(R1),@R2
```

3.6.2.2 Jump - The Jump command moves the pointer over the specified number of characters in the Text Buffer.

The form of the command is:

(+ or -)	nJ	Move the pointer (backward or forward) n characters.
	0J	Move the pointer to the beginning of the current line (equivalent to 0A).
	/J	Move the pointer to the end of the Text Buffer (equivalent to /A).
	=J	Move the pointer backward n characters, where n equals the length of the last text argument used.

Negative arguments move the pointer toward the beginning of the buffer, positive arguments toward the end. Jump treats carriage return, line feed, and form feed characters the same as any other character, counting one buffer position for each.

Examples:

*3J$$	Moves the pointer ahead three characters.
*-4J$$	Moves the pointer back four characters.
*B$GABC$=J$$	Move the pointer so that it immediately precedes the first occurrence of 'ABC' in the buffer.

3.6.2.3 Advance - The Advance command is similar to the Jump command except that it moves the pointer a specified number of lines (rather than single characters) and leaves it positioned at the beginning of the line.

The form of the command is:

nA	Advance the pointer forward n lines and position it at the beginning of the nth line.

-nA	Move the pointer backward past n <CR><LF> combinations and position it at the beginning of the -nth line.
0A	Advance the pointer to the beginning of the current line (equivalent to 0J).
/A	Advance the pointer to the end of the Text Buffer (equivalent to /J).

Examples:

*3A$$	Moves the pointer ahead three lines.

Assuming the buffer contains:

```
CLR     @R2
```

The command:

```
*0A$$
```

Moves the pointer to:

```
↑CLR     @R2
```

3.6.3 Search Commands

Search commands are used to locate specific characters or strings of characters within the Text Buffer.

3.6.3.1 Get - The Get command starts at the pointer and searches the current Text Buffer for the nth occurrence of a specified text string. If the search is successful, the pointer is left immediately following the nth occurrence of the text string. If the search fails, an error message is printed and the pointer is left at the end of the Text Buffer. The format of the command is:

```
nGtext$
```

The argument (n) must be positive and is assumed to be 1 if not otherwise specified. The text string may be any length and immediately follows the G command. The search is made on the portion of the text between the pointer and the end of the buffer.

Example:

Assuming the buffer contains:

```
↑MOV     PC,R1
 ADD     #DRIV-.,R1
 MOV     #VECT,R2
 CLR     @R2
 MOVB    5(R1),@R2
 ADD     R1,(R2)+
 CLR     @R2
 MOVB    6(R1),@R2
```

The command:

```
*GADD$$
```

positions the pointer at:

```
ADD.    #DRIV-.,R1
```

The command:

```
*3G@R2$$
```

positions the pointer at:

```
ADD     R1,(R2)+
CLR     @R2.
```

After search commands, the pointer is left immediately following the text object. Using a search command in combination with =J will place the pointer before the text object, as follows:

```
*GTEST$=J$$
```

This command combination places the pointer before 'TEST'.

3.6.3.2 Find - The Find command starts at the current pointer and searches the entire input file for the nth occurrence of the text string. If the nth occurrence of the text string is not found in the current buffer, a Next command is automatically performed and the search is continued on the new text in the buffer. When the search is successful, the pointer is left immediately following the nth occurrence of the text string. If the search fails (i.e., the end-of-file is detected for the input file and the nth occurrence of the text string has not been found), an error message is printed and the pointer is left at the beginning of an empty Text Buffer.

The form of the command is:

```
nFtext$
```

The argument (n) must be positive and is assumed to be 1 if not otherwise specified.

By deliberately specifying a nonexistent search string, the user can close out his file; that is, he can copy all remaining text from the input file to the output file.

Find is a combination of the Get and Next commands.

Example:

```
*2FMOVB 6(R1),@R2$$
```

Searches the entire input file for the second occurrence of the text string MOVB 6(R1),@R2. Each unsuccessfully searched buffer is written to the output file.

3.6.3.3 Position - The Position command searches the input file for the nth occurrence of the text string. If the desired text string is not found in the current buffer, the buffer is cleared and a new page is read from the input file. The format of the command is:

```
nPtext$
```

The argument (n) must be positive, and is assumed to be 1 if not otherwise specified. When a P command is executed the current contents of the buffer are searched from the location of the pointer to the end of the buffer. If the search is unsuccessful, the buffer is cleared and a new page of text is read and the cycle is continued.

If the search is successful, the pointer is positioned after the nth occurrence of the text. If it is not, the pointer is left at the beginning of an empty Text Buffer.

The Position command is a combination of the Get, Delete and Read commands; it is most useful as a means of placing the location pointer in the input file. For example, if the aim of the editing session is to create a new file from the second half of the input file, a Position search will save time.

The difference between the Find and Position commands is that Find writes the contents of the searched buffer to the output file while Position deletes the contents of the buffer after it is searched.

Example:

```
*PADD R1,(R2)+$$
```

Searches the entire input file for the specified string ignoring the unsuccessfully searched buffers.

3.6.4 Text Modification Commands

The following commands are used to insert, relocate, and delete text in the Text Buffer.

3.6.4.1 Insert - The Insert command causes the Editor to enter Text Mode and allows text to be inserted immediately following the pointer. Text is inserted until an ALTMODE is typed and the pointer is positioned immediately after the last character of the insert. The command format is:

```
Itext$
```

No arguments are used with the Insert command, and the text string is limited only by the size of the Text Buffer and the space available. All characters except ALTMODE are legal in the text string. ALTMODE terminates the text string.

NOTE

Forgetting to type the I command will cause the text entered to be executed as commands.

EDIT automatically protects against overflowing the Text Buffer during an Insert. If the I command is the first command in a multiple command line, EDIT ensures that there will be enough space for the Insert to be executed at least once. If repetition of the command exceeds the available memory, an error message is printed.

Example:

```
*IMOV    #BUFF,R2
MOV      #LINE,R1
MOVB     -1(R2),R0$$
*
```

Inserts the specified text at the current location of the pointer and leaves the pointer positioned after R0.

3.6.4.2 Delete - The Delete command removes a specified number of characters from the Text Buffer. Characters are deleted starting at the pointer; upon completion, the pointer is positioned at the first character following the deleted text.

The form of the command is:

(+ or -) nD	Delete n characters (forward or backward from the pointer).
0D	Delete from beginning of current line to the pointer (equivalent to 0K).
/D	Delete from pointer to end of Text Buffer (equivalent to /K).
=D	Delete -n characters, where n equals the length of the last text argument used.

Examples:

*-2D$$	Deletes the two characters immediately preceding the pointer.
*B$FMOV R1$=D$	Deletes the text string 'MOV R1'. (=D used in combination with a search command will delete the indicated text string).

Assuming a buffer of:

```
ADD       R1,(R2)+
CLR       ↑@R2
```

the command:

```
*0D$$
```

leaves the buffer with:

```
ADD       R1,(R2)+
↑@R2
```

3.6.4.3 Kill - The Kill command removes n lines from the Text Buffer. Lines are deleted starting at the location pointer; upon completion of the command, the pointer is positioned at the beginning of the line following the deleted text. The command format is:

nK	Delete lines beginning at the pointer and ending at the nth <CR><LF>.
-nK	Delete lines beginning with the first character in the -nth line and ending at the pointer.
0K	Delete from the beginning of the current line to the pointer (equivalent to 0D).
/K	Delete from the pointer to the end of the Text Buffer (equivalent to /D).

Example:

*2K$$	Delete lines starting at the current location pointer and ending at the 2nd <CR><LF>.

Assuming a buffer of:

```
ADD     R1,(R2)+
CLR     @R2
MOVB    6(R1),@R2
```

the command:

```
*/K$$
```

alters the contents of the buffer to:

```
ADD     R1,(R2)+
CLR
```

Kill and Delete commands perform the **same function**, except that Kill is line-oriented and Delete is **character-oriented.**

3.6.4.4 Change - The Change command **replaces n characters**, starting at the pointer, with the specified **text string** and leaves the pointer positioned immediately following the **changed text.**

The form of the command is:

(+ or -) nCtext$	Replace n characters (forward or backward from the pointer) with the specified text.
0Ctext$	Replace the characters from the beginning of the line up to the pointer with the specified text (equivalent to 0X).
/Ctext$	Replace the characters from the pointer to the end of the buffer with the specified text (equivalent to /X).

=Ctext$ Replace -n characters with the indicated text string, where n represents the length of the last text argument used.

The size of the text is limited only by the size of the Text Buffer and the space available. All characters are legal except ALTMODE which terminates the text string.

If the C command is to be executed more than once (i.e., it is enclosed in angle brackets) and if there is enough space available so that the command can be entered, it will be executed at least once (provided it appears first in the command string). If repetition of the command exceeds the available memory, an error message is printed. The Change command is identical to executing a Delete command followed by an Insert (nDItext$).

Examples:

*5C#VECT$$ Replaces the five characters to the right of the pointer with #VECT.

Assuming a buffer of:

```
CLR       @R2
MOV↑      5(R1),@R2
```

The command:

```
*0CADDB$$
```

leaves the buffer with:

```
CLR       @R2
ADDB↑     5(R1),@R2
```

=C can be used in conjunction with a search command to replace a specific text string as follows:

*GFIFTY:$=CFIVE:$ Find the occurrence of the text string FIFTY: and replace it with the text string FIVE:.

3.6.4.5 Exchange - The Exchange command exchanges n lines, beginning at the pointer, with the indicated text string and leaves the pointer positioned after the changed text.

The form of the command is:

nXtext$ Exchange all characters beginning at the pointer and ending at the nth <CR><LF> with the indicated text.

-nXtext$ Exchange all characters beginning with the first character on the -nth line and ending at the pointer with the indicated text.

0Xtext$ Exchange the current line from the beginning to the pointer with the specified text (equivalent to 0C).

/Xtext$ Exchange the lines from the pointer to the end of the buffer with the specifed text (equivalent to /C).

All characters are legal in the text string except ALTMODE which terminates the text.

The Exchange command is identical to a Kill command followed by an Insert (nKItext$), and accepts all legal line-oriented arguments.

If the X command is enclosed within angle brackets so that it will be executed more than once, and if there is enough memory space available so that the X command can be entered, it will be executed at least once (provided it is first in the command string). If repetition of the command exceeds the available memory, an error message is printed.

Example:

```
*2XADD  R1,(R2)+        Exchanges the two lines to
CLR     @R2             the right of the pointer location
$$                      with the text string.
*
```

3.6.5 Utility Commands

3.6.5.1 Save - The Save command starts at the pointer and copies the specified number of lines into the Save Buffer (described previously in Section 3.5).

The form of the command is:

nS

The argument (n) must be positive. The pointer position does not change and the contents of the Text Buffer are not altered. Each time a Save is executed, the previous contents of the Save Buffer, if any, are destroyed. If the Save command causes an overflow of the Save Buffer, an error message is printed.

Example:

Assume the Text Buffer contains the following assembly language subroutine:

```
;SUBROUTINE MSGTYP
;WHEN CALLED, EXPECTS R0 TO POINT TO AN
;ASCII MESSAGE THAT ENDS IN A ZERO BYTE,
;TYPES THAT MESSAGE ON THE USER TERMINAL

                .ASECT
                .=1000
MSGTYP:         TSTB (%0)           ;DONE?
                BEQ MDONE           ;YES-RETURN
MLOOP:          TSTB @#177564       ;NO-IS TERMINAL READY?
                BPL MLOOP           ;NO-WAIT
                MOVB (%0)+,@#177566 ;YES PRINT CHARACTER
                BR MSGTYP           ;LOOP
MDONE:          RTS %7              ;RETURN
```

The command:

```
*145$$
```

stores the entire subroutine in the Save Buffer; it may then be inserted in a program wherever needed by using the U command.

3.6.5.2 Unsave - The Unsave command inserts the entire contents of the Save Buffer into the Text Buffer at the pointer location and leaves the pointer positioned following the inserted text.

The form of the command is:

U	Insert in the Text Buffer the contents of the Save Buffer.
0U	Clear the Save Buffer and reclaim the area for text.

Zero is the only legal argument to the U command.

The contents of the Save Buffer are not destroyed by the Unsave command (only by the 0U command) and may be Unsaved as many times as desired.

If there is no text in the Save Buffer and the U command is given, the ?*NO TEXT*? error message is printed. If the Unsave command causes an overflow of the Text Buffer, the ?*NO ROOM*? error message is displayed.

3.6.5.3 Macro - The Macro command inserts a command string into the EDIT Macro Buffer. The Macro command is of the form:

M/command string/	Store the command string in the Macro Buffer.
0M	Clear the Macro Buffer and reclaim the area for text.
or	
M//	

/ represents the delimiter character. The delimiter is always the first character following the M command, and may be any character which does not appear in the Macro command string itself.

Starting with the character following the delimiter, EDIT places the Macro command string characters into its internal Macro Buffer until the delimiter is encountered again. At this point, EDIT returns to Command Mode. The Macro command does not execute the Macro string; it merely stores the command string so that it can be executed later by the Execute Macro (EM) command. Macro does not affect the contents of the Text or Save Buffers.

All characters except the delimiter are legal Macro command string characters, including single ALTMODEs to terminate text commands. All commands, except the M and EM commands, are legal in a command string macro.

In addition to the 0M command, typing the M command immediately followed by two identical characters (assumed to be delimiters) and two ALTMODE characters also clears the Macro Buffer.

Examples:

*M//$$	Clears the Macro Buffer
*M/GR0$-C1$/$$	Stores a Macro to change R0 to R1.

NOTE

Be careful to choose infrequently used characters as macro delimiters; use of frequently used characters can lead to inadvertent errors. For example,

```
*M GMOV R0$=CADD R1$ $$
?*NO FILE*?
```

In this case, it was intended that the macro be GMOV R0$=CADD R1$ but since the delimiter character (the character following the M) is a space, the space following MOV is used as the second delimiter, terminating the macro. EDIT then returns an error when the R0$= becomes an illegal command structure.

3.6.5.4 Execute Macro - The Execute Macro command executes the command string specified in the last Macro command.

The form of the command is:

nEM

The argument (n) must be positive. The macro is executed n times and returns control to the next command in the original command string.

Examples:

```
*M/BGR0$-C1$/$$
*B1000EM$$
?*SRCH FAIL IN MACRO*?
*
```

Executes the MACRO stored in the previous example. An error message is returned when the end of buffer is reached. (This macro effectively changes all occurrences of R0 in the Text Buffer to R1.)

```
*IMOV PC,R1$2EMICLR @R2$$
*
```

In a new program, inserts MOV PC,R1 then executes the command in the Macro Buffer twice before inserting CLR @R2.

3.6.5.5 Edit Version - The Edit Version command displays the version number of the Editor in use on the console terminal.

The form of the command is:

```
EV$
```

Example:

```
*EV$$
V02-01
*
```

3.6.5.6 Upper- and Lower-Case Commands - Users who have any upper/lower-case terminal as part of their hardware configuration may take advantage of the upper- and lower-case capability of this terminal. Two editing commands, EL and EU, permit this.

When the Editor is first called (R EDIT), upper-case mode is assumed; all characters typed are automatically translated to upper case. To allow processing of both upper- and lower-case characters, the Edit Lower command is entered:

```
*EL$$
*i Text and commands can be entered in UPPER and lower case.$$
*
```

The Editor now accepts and echoes upper- and lower-case characters received from the keyboard, and outputs text on the teleprinter in upper- and lower-case.

To return to upper-case mode, the Edit Upper command is used:

```
*EU$$
```

Control also reverts to upper-case mode upon exit from the Editor (via EF, EX, or CRTL C).

Note that when an EL command has been issued, Edit commands can be entered in either upper- or lower-case. Thus, the following two commands are equivalent:

```
*@TEXT$=Cnew text$V$$

*sTEXT$=cnew text$v$$
```

The Editor automatically translates (internally) all commands to upper-case independent of EL or EU.

3.7 THE DISPLAY EDITOR

In addition to all functions and commands mentioned thus far, the Editor has additional capabilities to allow efficient use of VT-11 display hardware which may be part of the system configuration (GT40, GT44, DECLAB 11/40).

The most apparent feature is the ability to use the display screen rather than the console terminal as a window into the Text Buffer for printout of all textual input and output. When all the features of the display Editor are in use, a 12" screen displays text as shown in Figure 3-1:

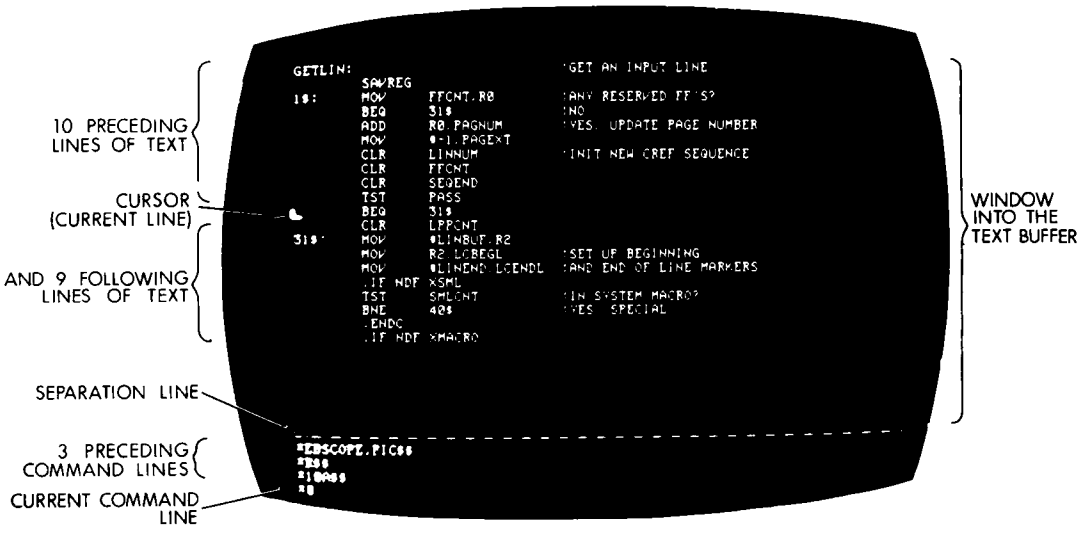


Figure 3-1
Display Editor Format

The major advantage is that the user can now see immediately where the pointer is. The pointer appears between characters on the screen as a bright blinking L-shaped cursor and can be detected easily and quickly. Note that if the pointer is placed between a carriage return and line feed, it appears in an inverted position at the beginning of the next line.

In addition to displaying the current line (the line containing the cursor), the 10 lines of text preceding the current line and the 9 lines following it are also in view. Each time a command string is executed (via a double ALTMODE) this portion of the screen is refreshed so that it reflects the results of the commands just performed.

The lower section of the screen contains 4 lines of editing commands. The command line currently being entered is last, preceded by the three most recent command lines. This section is separated from the text portion of the screen by a horizontal line of dashes. As new command lines are entered, previous command lines are scrolled upward off the command section so that only four command lines are ever in view.

A 17" screen displays 30 lines of text and 8 command lines.

3.7.1 Using the Display Editor

The display features of the Editor are automatically invoked whenever the system scroller is in use and the user types:

```
R EDIT
```

However, if the system does not contain VT-11 display hardware, the display features are not enabled.

Providing that the system does contain VT-11 display hardware and that the user wishes to employ the screen during the editing session, he may activate it in one of two ways (all editing commands and functions previously discussed in this chapter are valid for use):

1. If the scroller is in use (i.e., the GT ON monitor command has been typed prior to calling the Editor), EDIT recognizes this and automatically continues using the screen for display of text and commands. However, it rearranges the scroller so that a "window" into the Text Buffer appears in the top two/thirds of the screen, while the bottom third is used to display command lines. This arrangement is shown in Figure 3-1.

   The Edit Console command can be used to return the scroller to its normal mode so that text and commands appear as described in Chapter 2, Section 2.7.1 (i.e., using the full screen for display of command lines, and eliminating the window). The form of the command is:

   ```
   EC
   ```

For example:

*BAEC2L$$	The second and third lines of the current buffer are listed on the screen; there is no window into the Text Buffer at this point.

Subsequent EC commands are ignored if the window into the Text Buffer is not being displayed.

To recall the window, the Edit Display command is used:

```
ED
```

The screen is again arranged as shown in Figure 3-1.

2. Assume the scroller is not in use (i.e., the GT ON command has not been typed, or the monitor GT OFF command has been typed prior to calling the Editor). When the user calls EDIT, an asterisk appears on the console terminal as described in Section 3.1. Using the ED command at this time provides the window into the Text Buffer; however, commands continue to be echoed to the console terminal.

   When ED is used in this case, it must be the first command issued; otherwise, it becomes an illegal command (since the memory used by the display buffer and code, amounting to over 600 words, is reclaimed as working space). The display cannot be used again until a fresh copy of EDIT is loaded.

   While the display of the text window is active, ED commands are ignored.

   Typing the EC command clears the screen and returns all output to the console terminal.

NOTE

Under the Single-Job Monitor only, after the editing session is over, it is recommended that the screen be cleared by either typing the EC command, or returning to the monitor and using the monitor INITIALIZE command. Failure to do this may cause unpredictable results.

3.7.2 Setting the Editor to Immediate Mode

An additional mode is available in EDIT to provide an easier and faster degree of interaction during the editing session. This mode is called Immediate Mode and combines the most-used functions of the Text and Command Modes--namely, to reposition the pointer and to delete and insert characters.

Immediate Mode may be used only when the VT-11 display hardware is active and the Editor is running; it is entered by typing two ALTMODES (only) in response to the Command Mode asterisk:

```
*$$
```

The Editor responds by echoing an exclamation point on the screen. The exclamation character remains on the screen as long as control remains in Immediate Mode.

Once Immediate Mode has been entered, only the commands in Table 3-3 are used. None of these commands echoes, but the text appearing on the screen is constantly refreshed and updated during the editing process. Note that no EDIT commands other than those in Table 3-3 may be used while control remains in Immediate Mode.

To return control to the display Editor's normal Command Mode at any time while in Immediate Mode, type a single ALTMODE. The Editor responds with an asterisk and the user may proceed using all normal Editing commands. (Immediate Mode commands typed at this time will be accepted as Command Mode input characters.) To return control to the monitor while in Immediate Mode, type CTRL C.

Table 3-3
Immediate Mode Commands

Command	Meaning
CTRL N	Advance the pointer (cursor) to the beginning of the next line (equivalent to A).
CTRL G	Move the pointer (cursor) to the beginning of the previous line (equivalent to -A).
CTRL D	Move the pointer (cursor) forward by one character (equivalent to J).
CTRL V	Move the pointer (cursor) back by one character (equivalent to -J).
RUBOUT	Delete the character immediately preceding the pointer (cursor) (equivalent to -D).
CTRL C	Return control to the monitor.
ALTMODE (one only) (two)	Return control to Command Mode. Direct control to Immediate Mode.
Any other character than those above	Insert the character as text positioned immediately before the pointer (cursor) (equivalent to I).

3.8 EDIT EXAMPLE

The following example illustrates the use of some of the EDIT commands to change a program stored on the device DK. Sections of the terminal output are coded by letter and corresponding explanations follow the example.

```
A  .R EDIT
   *ERDK:TEST1.MAC$$
   *EWDK:TEST2.MAC$$
   *R$$
B  *#L$$
   ;TEST PROGRAM

   START:  MOV #1000,%6     ;INITIALIZE STACK
           MOV #MSG,%0      ;POINT R0 TO MESSAGE
           JSR %7,MSGTYP    ;PRINT IT
           HALT             ;STOP
   MSG:    .ASCII/IT WORKS/
           .BYTE 15
           .BYTE 12
           .BYTE 0
C  *B 1J 5D$$
D  *GPROGRAM$$
   *0L$$
E  ;PROGRAM*I TO TEST SUBROUTINE MSGTYP  TYPES
   ;"THE TEST PROGRAM WORKS"
   ;ON THE TEMI\IM\RMINAL$$
F  *F.ASCII/$$
   *8CTHE TEST PROGRAM WORKS$$
G  *P.BYTE^X
   *F.BYTE 0$V$$
           .BYTE 0
H  *I
           .END
   $B/L$$
   ;PROGRAM TO TEST SUBROUTINE MSGTYP.  TYPES
   ;"THE TEST PROGRAM WORKS"
   ;ON THE TERMINAL

   START:  MOV #1000,%6     ;INITIALIZE STACK
           MOV #MSG,%0      ;POINT R0 TO MESSAGE
           JSR %7,MSGTYP    ;PRINT IT
           HALT             ;STOP
   MSG:    .ASCII/THE TEST PROGRAM WORKS/
           .BYTE 15
           .BYTE 12
           .BYTE 0
           .END
I  *EX$$
   .
```

A The EDIT program is called and prints an *. The input file is TEST1.MAC; the output file is TEST2.MAC and the first page of input is read.

B The buffer contents are listed.

C Be sure the pointer is at the beginning of the buffer. Advance pointer one character (past the ;) and delete the "TEST ".

D Position pointer after PROGRAM and verify the position by listing up to the pointer.

E Insert text. RUBOUT used to correct typing error.

F Search for .ASCII/ and change "IT WORKS" to "THE TEST PROGRAM WORKS".

G CTRL X typed to cancel P command. Search for ".BYTE 0" and verify location of pointer with V command.

H Insert text. Return pointer to beginning of buffer and list entire contents of buffer.

I Close input and output files after copying the current text buffer as well as the rest of input file into output file. EDIT returns control to the monitor.

3.9 EDIT ERROR MESSAGES

The Editor prints an error message whenever one of the error conditions listed next occurs. Prior to executing any commands, the Editor first scans the entire command string for errors in command format (illegal arguments, illegal combinations of commands, etc.). If an error of this type is found, an error message of the form:

```
?ERROR MSG?
```

is printed and no commands are executed. The user must retype the command.

If the command string is syntactically correct, execution is started. Execution errors are still possible, however (buffer overflow, I/O errors, etc.), and if such an error occurs, a message of the form:

```
?*ERROR MSG*?
```

is printed. In this case, all commands preceding the one in error are executed, while the command in error and those following are not executed. Most errors will generally be of the syntax type and can be corrected before execution.

When an error occurs during execution of a Macro, the message format is:

?message IN MACRO?

or

?*message IN MACRO*?

depending on when it is detected.

Message	Explanation
CB ALMOST FULL	The command currently being entered is within 10 characters of exceeding the space available in the Command Buffer.
?CB FULL?	Command exceeds the space allowed for a command string in the Command Buffer.
?*DIR FULL*?	No room in device directory for output file.
?*EOF*?	Attempted a Read, Next or file searching command and no data was available.
?*FILE FULL*?	Available space for an output file is full. Type a CTRL C and the CLOSE monitor command to save the data already written.
?*FILE NOT FND*?	Attempted to open a nonexisting file for editing.
?*HDW ERR*?	A hardware error occurred during I/O. May be caused by WRITE LOCKed device. Try again.
?ILL ARG?	The argument specified is illegal for the command used. A negative argument was specified where a positive one was expected or argument exceeds the range + or - 16,383.
?ILL CMD?	EDIT does not recognize the command specified; ED was not the first command issued when used to activate the display hardware.
?*ILL DEV*?	Attempted to open a file on an illegal device, or attempted to use display hardware when none was available (it may be in use by the other job).
?ILL MAC?	Delimiters were improperly used, or an attempt was made to enter an M command during execution of a Macro or an EM command while an EM was in progress.

Text Editor

Message	Explanation
?*ILL NAME*?	File name specified in EB, EW, or ER is illegal.
?*NO FILE*?	Attempted to read or write when no file is open.
?*NO ROOM*?	Attempted to Insert, Save, Unsave, Read, Next, Change or Exchange when there was not enough room in the appropriate buffer. Delete unwanted buffers to create more room or write text to the output file.
?*NO TEXT*?	Attempted to call in text from the Save Buffer when there was no text available.
?*SRCH FAIL*?	The text string specified in a Get, Find or Position command was not found in the available data.
?"<>"ERR?	Iteration brackets are nested too deeply or used illegally or brackets are not matched.

DIGITGRAPH
P. O. Box 4249
Tucson, Arizona 85717

SOFTWARE PROBLEMS OR ENHANCEMENTS

Questions, problems, and enhancements to Digital software should be reported on a Software Performance Report (SPR) form and mailed to the SPR Center at one of the following Digital Offices: (SPR forms are available from the SPR Center.)

Areas Covered	SPR Center
Australia/New Zealand	Digital Equipment Australia Pty. Ltd. 123-125 Willoughby Road, P.O. Box 491 Crows Nest New South Wales, Australia 2065
Brazil	Digital Equipment Comercio E Industria LTDA Rua Batatais, 429 (Esq. Al. Campinas) 01423-Jardim Paulista São Paulo-SP-Brazil
Canada	Digital Equipment of Canada, Ltd. Software Services P.O. Box 11500, K2H 8K8 Ottawa, Ontario, Canada
Caribbean	Digital Equipment Latin America, Inc. 407 del Parque Street Santurce, Puerto Rico 00912
France	Digital Equipment France 18, rue Saarinen Centre Silic - CIDEX L225 F-94533 Rungis, France
Israel	DEC-sys Computers Ltd. 7 Habakuk Street IL-Tel Aviv 63505, Israel
Italy	Digital Equipment S.P.A. Corso Garibaldi 49 I-20121 Milano, Italy
Japan	Digital Equipment Corp. Intl. Kowa Building 25 (3d Floor) 8-7 Sunban-Cho Chiyoda-ku, Tokyo 102, Japan
Mexico	Equipo Digital, S.A. de C.V. 109 Concepcion Beistegui Mexico 12, D.F.
The Netherlands	Digital Equipment B.V. Kaap Hoorndreef 38, P.O. Box 9064 NL-Utrecht - Overvecht, The Netherlands
Nordic	Digital Equipment AB Englundavägen 7 S-17141 Solna Sweden
Switzerland Spain Portugal Greece Bulgaria Romania Yugoslavia	Digital Equipment Corp. SA 20, Quai Ernest Ansermet Case Postale 23, CH-1211 Geneva 8 Switzerland
United Kingdom	Digital Equipment Co. Ltd. Fountain House, Butts Centre GB-Reading RG1 7QN, England
West Germany Austria East Germany Russia Hungary Poland Czechoslovakia	Digital Equipment GmbH D-8000 München 40 Wallensteinplatz 2 West Germany
United States; remainder of Far East, Middle East, Africa, Latin America	Software Communications P.O. Box F Maynard, MA 01754

DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
European Headquarters: 81 route de l'Aire, 1211 Geneva 26. Switzerland
Digital Equipment of Canada Ltd., P.O. Box 11500 Ottawa, Ontario K2H8K8.

DIGITAL EQUIPMENT CORPORATION, Component Group Headquarters: 1 Iron Way, Marlborough, Mass. 01752, Telephone: (617) 481-7400

DIGITAL EQUIPMENT CORPORATION, Corporate Headquarters: Maynard, Massachusetts 01754, Telephone: (617) 897-5111

SALES AND SERVICE OFFICES

DOMESTIC – ARIZONA, Phoenix and Tucson • CALIFORNIA, Los Angeles, Monrovia, Oakland, Ridgecrest, San Diego, San Francisco (Mountain View), Santa Ana, Sunnyvale, and Woodland Hills • COLORADO, Englewood • CONNECTICUT, Fairfield and Meriden • DISTRICT OF COLUMBIA, Washington (Latham, Md.) • FLORIDA, Orlando • GEORGIA, Atlanta • ILLINOIS, Chicago (Rolling Meadows) • INDIANA, Indianapolis • IOWA, Bettendorf • KENTUCKY, Louisville • LOUISIANA, Metairie (New Orleans) • MASSACHUSETTS, Marlborough and Waltham • MICHIGAN, Detroit (Farmington Hills) • MINNESOTA, Minneapolis • MISSOURI, Kansas City and St. Louis • NEW HAMPSHIRE, Manchester • NEW JERSEY, Fairfield, Metuchen and Princeton • NEW MEXICO, Albuquerque • NEW YORK, Albany, Huntington Station, Manhattan, Rochester and Syracuse • NORTH CAROLINA, Durham/Chapel Hill • OHIO, Cleveland, Columbus and Dayton • OKLAHOMA, Tulsa • OREGON, Portland • PENNSYLVANIA, Philadelphia (Bluebell) and Pittsburgh • TENNESSEE, Knoxville • TEXAS, Austin, Dallas and Houston • UTAH, Salt Lake City • WASHINGTON, Bellevue • WISCONSIN, Milwaukee (Brookfield) • INTERNATIONAL – ARGENTINA, Buenos Aires • AUSTRALIA, Adelaide, Brisbane, Canberra, Melbourne, Perth and Sydney • AUSTRIA, Vienna • BELGIUM, Brussels • BOLIVIA, La Paz • BRAZIL, Puerto Alegre, Rio de Janeiro and São Paulo • CANADA, Calgary, Halifax, Montreal, Ottawa, Toronto and Vancouver • CHILE, Santiago • DENMARK, Copenhagen • FINLAND, Helsinki • FRANCE, Grenoble and Paris • GERMANY, Berlin, Cologne, Hannover, Hamburg, Frankfurt, Munich and Stuttgart • HONG KONG • INDIA, Bombay • INDONESIA, Djakarta • ISRAEL, Tel Aviv • ITALY, Milan and Turin • JAPAN, Osaka and Tokyo • MALAYSIA, Kuala Lumpur • MEXICO, Mexico City • NETHERLANDS, Utrecht • NEW ZEALAND, Auckland • NORWAY, Oslo • PHILIPPINES, Manila • PUERTO RICO, Santurce • SINGAPORE • SPAIN, Barcelona and Madrid • SWEDEN, Gothenburg and Stockholm • SWITZERLAND, Geneva and Zurich • TAIWAN, Taipei and Taoyuan • UNITED KINGDOM, Birmingham, Bristol, Dublin, Edinburgh, Leeds, London, Manchester and Reading • VENEZUELA, Caracas • YUGOSLAVIA, Ljubljana •